

# **Selezione ed Iterazione**

## **Lezione 5**

# Scopo della Lezione

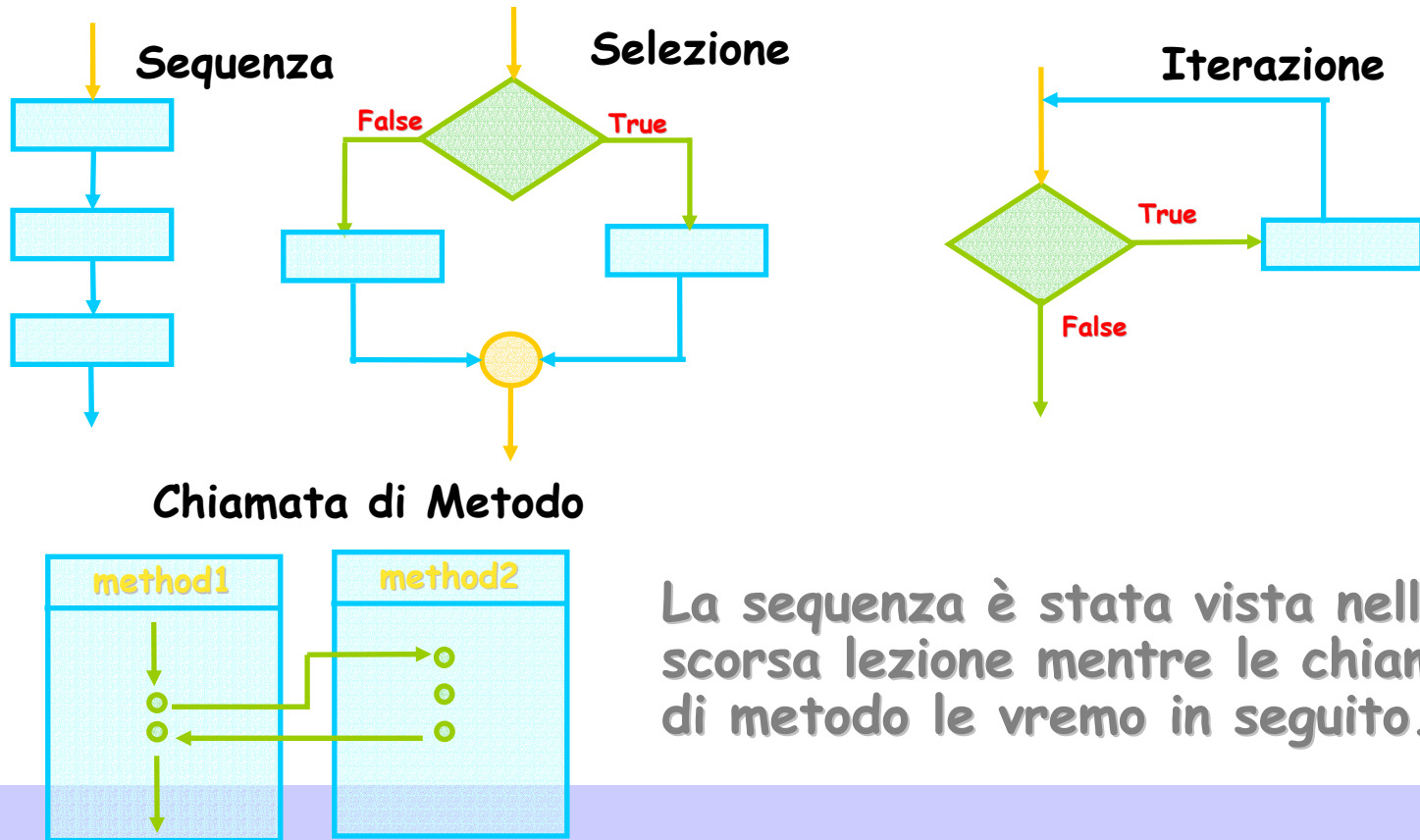
- Ripassare i concetti di selezione e iterazione;
- Dare nozioni di programmazione strutturata;
- Verificare come uno stesso programma possa essere scritto in forme differenti.

# Programmazione Strutturata

- Programmazione Strutturata: sfrutta un piccolo insieme di strutture di controllo predefinite.
  - **Sequenza**. Le istruzioni di un programma sono eseguite in ordine sequenziale a meno che il loro flusso non sia interrotto da una delle seguenti strutture di controllo.
  - **Selezione**. if, if-else e switch sono istruzioni di selezione che permettono di biforcare il flusso di controllo scegliendo tra 2 o più alternative.
  - **Iterazione**. for, while e do-while sono istruzioni di controllo cicliche che permettono al programma di ripetere una sequenza di istruzioni.
  - **Invocazione di Metodo**. Invocando un metodo, il controllo è temporaneamente trasferito al metodo invocato e ritorna all'invocante quando l'esecuzione del metodo è terminata.

# Programmazione Strutturata: Costrutti

Non importa quanto grande sia il programma, il suo flusso di controllo potrà sempre essere espresso come combinazione di questi quattro costrutti.

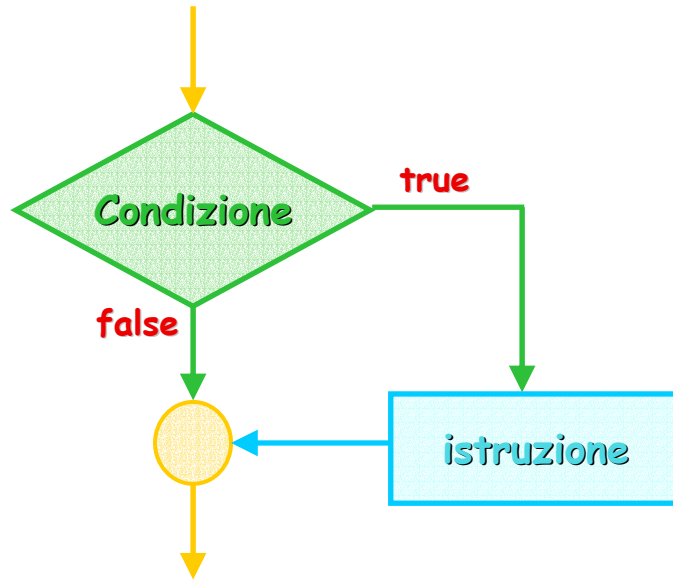


La sequenza è stata vista nella scorsa lezione mentre le chiamate di metodo le vremo in seguito.

# Flusso di Controllo

- Il flusso di esecuzione del programma è gestito da istruzioni di controllo che permettono di scegliere un cammino tra molti.
- Il cammino da seguire viene scelto in base al soddisfacimento di alcune condizioni.

# Flusso di Controllo: If



if (espressione boolean)  
istruzione

Se l'espressione booleana che rappresenta la condizione dell'if è valutata true, allora sarà eseguita "istruzione". Altrimenti "istruzione" non verrà eseguita.

# Flusso di Controllo: Espressioni Booleane

- Le espressioni booleane sono espressioni che assumono valori booleani, cioè valgono vero o falso.
- Esempi di espressioni booleane:
  - `true`                      `false`
  - `isSleeping`                `(1 + 1) == 2`
- `==` è l'operatore di uguaglianza in Java, `!=` è la negazione dell'operatore di uguaglianza (cioè l'operatore di disuguaglianza)

# Flusso di Controllo: Espressioni Booleane (Segue)

Tavola di verità per gli operatori booleani: AND (&&), OR (||), OR-ESCLUSIVO (^) e NOT (!).

Oper <sub>1</sub>	Oper <sub>2</sub>	AND	OR	XOR	NOT
$o_1$	$o_2$	$o_1 \&\& o_2$	$o_1    o_2$	$o_1 \wedge o_2$	$!o_1$
true	true	true	true	false	false
true	false	false	true	true	false
false	true	false	true	true	true
false	false	false	false	false	true

I dati boolean hanno solo due possibili valori: true e false.

$o_1 || o_2$  è true se uno dei due operandi è true.

$o_1 \&\& o_2$  è true solo se entrambi  $o_1$  e  $o_2$  sono true.

$!o_1$  è true quando  $o_1$  è false.

$o_1 \wedge o_2$  è true se solo uno tra  $o_1$  e  $o_2$  è true.

# Flusso di Controllo: Espressioni Booleane (Precedenze)

Ordine di precedenza degli operatori booleani.

Ordine di Precedenza	Operatore	Operazione
1	( )	Parentesi
2	!	NOT
3	&&	AND
4	^	XOR
5		OR

In un'espressione mista, la valutazione del NOT precede quella dell'AND, che precede quella dell'XOR, che precede quella dell'OR.

AND è valutato prima dell'OR perché ha una precedenza più alta.

## ESPRESSIONE

true || true && false

(true || true) && false  
true || (true && false)

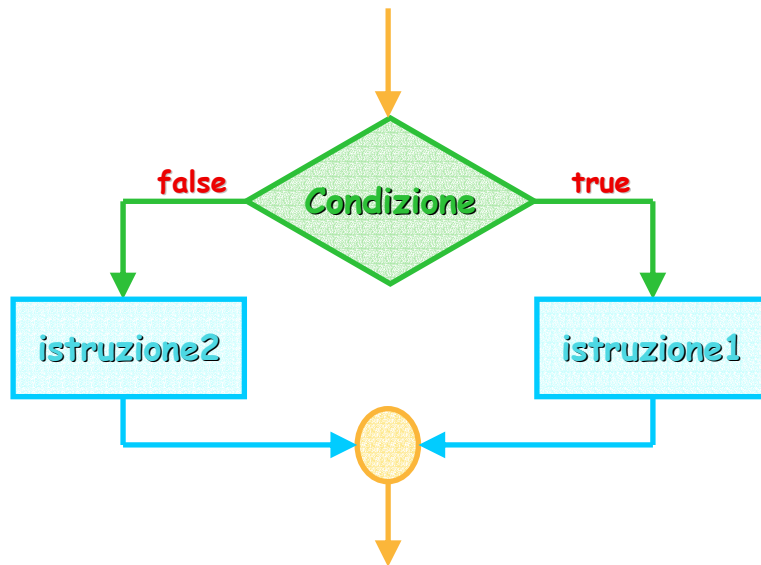
## VALUTAZIONE

true || false → true

true && false → false  
true || false → true

Le parentesi possono annullare la relazione di precedenza.

# Flusso di Controllo: If-Then-Else

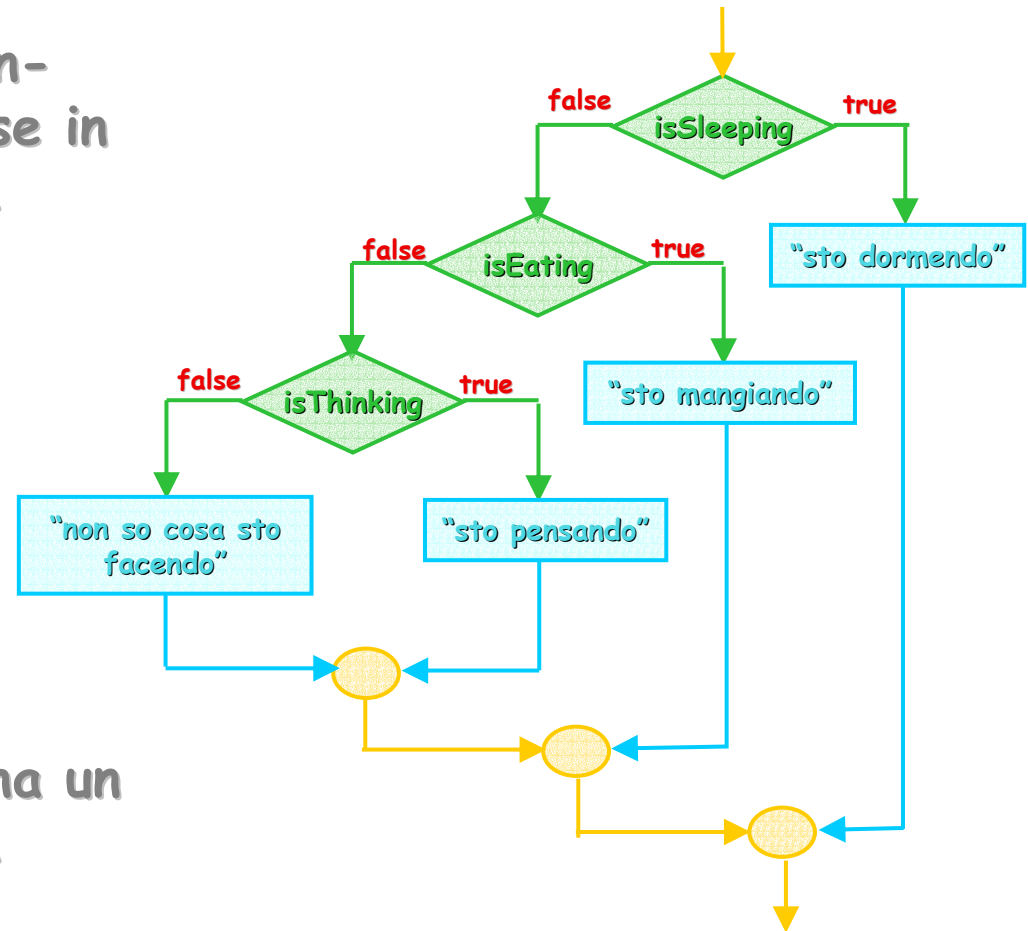


```
if (espressione booleana)  
    istruzione1;  
else istruzione2;
```

- Se la condizione è vera si esegue "istruzione1" altrimenti viene eseguita "istruzione2".

# Flusso di Controllo: Selezione Multipla

Diverse istruzioni if-then-else possono essere messe in cascata per formare una struttura di selezione multipla.



Nota: questa struttura ha un solo ingresso ed una sola uscita.

# Flusso di Controllo: Switch

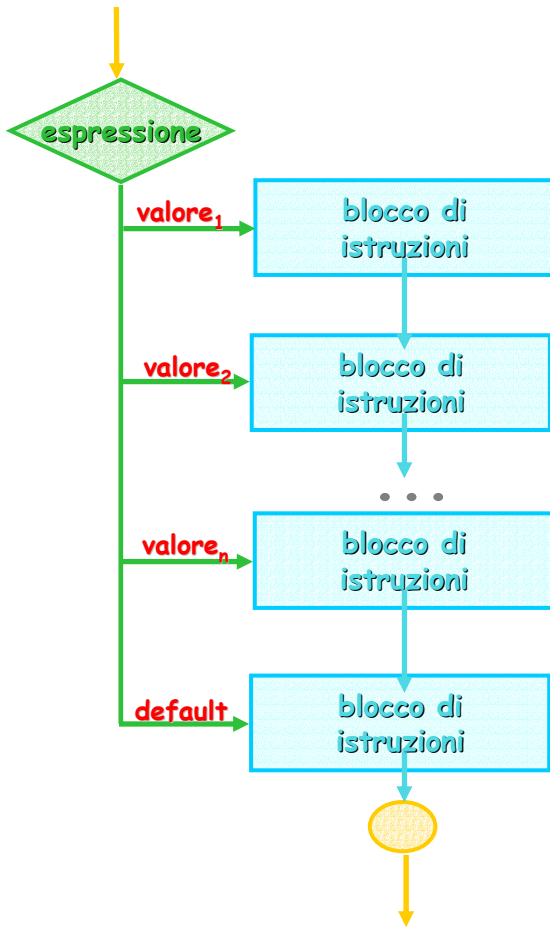
Sintassi:

```
switch (espressione) {  
  case valore1: blocco di istruzioni  
  case valore2: blocco di istruzioni  
  default: blocco di istruzioni  
}
```

Semantica:

- valuta l'espressione.
- passa il controllo (switch) o all'etichetta case il cui valore corrisponde a quello ottenuto valutando espressione oppure alla clausola default (quando nessuna etichetta corrisponde al valore calcolato).
- esegue tutte le istruzioni fino alla fine dello switch.

# Flusso di Controllo: Switch



Esempio:

```

int m = 2;
switch (m) {
  case 1:
    System.out.println("m = 1");
  case 2:
    System.out.println("m = 2");
    break;
  case 3:
    System.out.println("m = 3");
    break;
  default:
    System.out.println("caso di default");
}
  
```

l'istruzione break trasferisce il controllo fuori dallo switch.

# Flusso di Controllo: Esempi

```
if (isEating)
    return "Sta mangiando";
```

if

if-then-else

```
if (isEating)
    System.out.println("Sta mangiando");
else
    System.out.println("Non sta mangiando");
```

```
if (isSleeping)
    System.out.println("Sta dormendo");
else if (isEating)
    System.out.println("Sta mangiando");
else if (isThinking)
    System.out.println("Sta pensando");
else
    System.out.println("Errore: non so che sta facendo");
```

Selezione  
Multipla

# Flusso di Controllo: Il Problema dell'else Pendente

- Il programmatore deve stare attento ad associare ogni "else" con il corrispondente "if".
- Regola: ogni "else" è associato all'"if" più vicino non ancora associato.
- L'indentazione (che il compilatore ignora) dovrebbe riflettere la logica dell'istruzione.

## Indentazione Errata

```
if (condition1)
    if (condition2)
        System.out.println("One");
else
    System.out.println("Two");
```

## Indentazione Corretta

```
if (condition1)
    if (condition2)
        System.out.println("One");
else
    System.out.println("Two");
```

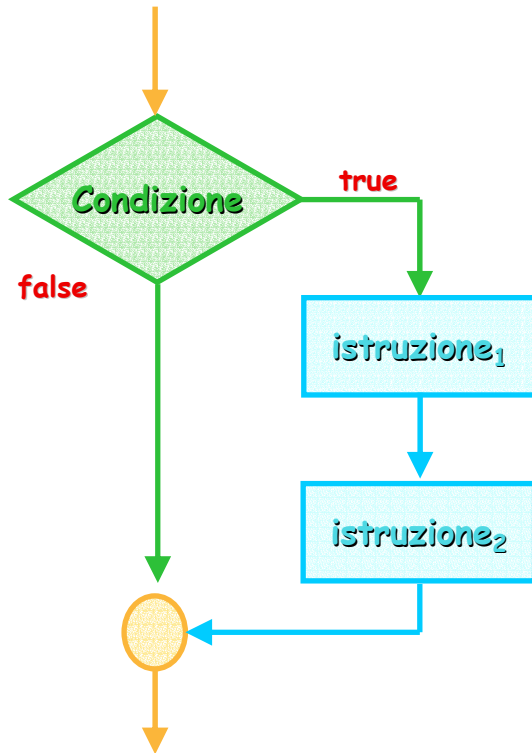
# Flusso di Controllo: Il Problema dell'else Pendente

- Nel caso si voglia eludere questa regola è necessario utilizzare le parentesi graffe

```
if (condition1) {  
    if (condition2)  
        System.out.println("One");  
} else  
    System.out.println("Two");
```

# Flusso di Controllo: Blocchi di Istruzioni

- Le parentesi graffe possono essere usate anche per eseguire più istruzioni all'interno di una selezione if



```
if (espressione booleana) {  
    istruzione1;  
    istruzione2;  
}
```

# Flusso di Controllo: Iterazione

Iterazione: struttura di controllo pensata per ripetere una sequenza di istruzioni.

Se il numero di iterazioni è noto si usa un ciclo numerato:

- Contare il numero di volte che la lettera 'a' ricorre in un documento:

```
Inizializza totalAs a 0
per ogni carattere nel documento
  se il carattere è una 'a'
    aggiungi 1 a totalAs
restituisce totalAs come risultato
```

- Stampare i numeri tra 1 e 5000:

```
per ogni numero, N, da 1 a 5000
  stampa N
```

# Flusso di Controllo: Iterazione

Se il numero di iterazioni è sconosciuto, si usa un ciclo condizionato.

- Cercare in un file la scheda di uno studente:

ripeti i seguenti passi  
    leggi una scheda da file  
finché non viene letto la scheda di Paolo Rossi

- Calcolare il numero medio di orsi avvistati mensilmente:

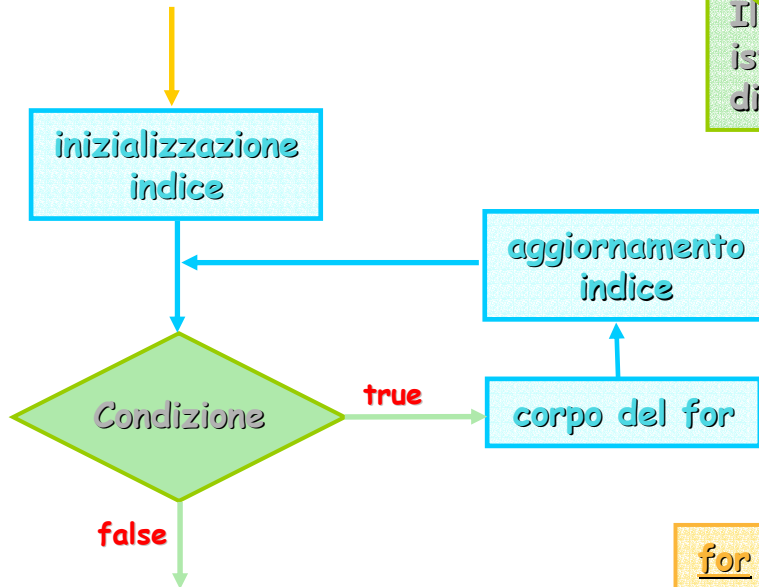
Inizializzare `sumOfBears` e `numOfMonths` a 0  
Ripeti i seguenti passi  
    leggi un numero dalla tastiera  
    aggiungilo a `sumOfBears`  
    aggiungi 1 a `numOfMonths`  
Finché l'utente non vuole fermarsi  
dividi `sumOfBears` per `numOfMonths`  
restituisce la media

# Flusso di Controllo: For

Sintassi:

```
for ( inizializzazione indice; condizione ; aggiornamento indice )  
    corpo del for;
```

Semantica:



Il corpo del for può essere una singola istruzione o un blocco (cioè una sequenza di istruzioni racchiuse tra {}).

```
for (int k = 0; k < 100; k++) // per 100 volte  
    System.out.println("ciao"); // stampa "ciao"
```

# Flusso di Controllo: For

Se `k` è dichiarata nell'istruzione `for`, non può essere usata fuori dal ciclo:

```
for (int k = 0; k < 100; k++)  
    System.out.println("Ciao");  
System.out.println("k = " + k); // Errore, k non è stata dichiarata
```

Se la variabile `k` è stata dichiarata prima dell'istruzione `for`, allora può essere usata anche dopo il ciclo:

```
int k = 0; // dichiarazione dell'indice  
for (k = 0; k < 100; k++)  
    System.out.println("Ciao");  
System.out.println("k = " + k); // uso corretto
```

# Flusso di Controllo: For

Il ciclo for parte inizializzando il proprio indice ad un valore iniziale e poi conta 0 o più iterazioni finché il limite prefissato non viene raggiunto.

La condizione verifica se il limite è stato raggiunto.

```
public void countdown() {  
    for (int k = 10; k > 0; k--)  
        System.out.print(k + " ");  
  
    System.out.println("BOOOM!!!!!!");  
} // countdown()
```

L'aggiornamento dell'indice lo incrementa fino al limite.

**Cicli infiniti:** un ciclo che non riesce a raggiungere il proprio limite. Ad es. si ha un ciclo infinito incrementando k.

# Flusso di Controllo: For

I cicli for possono essere annidati. La seguente tabella mostra la relazione tra gli indici dei cicli annidati.

```
## ## ## ## ##
## ## ## ##
## ## ##
## ##
##
```

Riga	Colonne (6 - Riga)
1	6-1 = 5
2	6-2 = 4
3	6-3 = 3
4	6-4 = 2
5	6-5 = 1

```
for (int row = 1; row <= 5; row++) { // per ogni riga
    for (int j = 1; j <= 6 - row; j++) // stampa la riga
        System.out.print('#');
    System.out.println(); // e vai a capo
}
```

# Flusso di Controllo: While

Es. Il problema  $3N + 1$ . Se  $N$  è un intero positivo allora la sequenza generata dalla seguente regola terminerà sempre con 1:

Caso	Operazione
$N$ dispari	$N = 3 * N + 1$
$N$ pari	$N = N / 2$

Ad esempio:

10, 5, 16, 8, 4, 2, 1

Non conoscendo a priori il numero di iterazioni non si può (per lo meno è molto difficile) usare un ciclo for:

```

N = 50;
while (N != 1) {
    System.out.print(N + " ");
    if (N % 2 == 0)
        N = N / 2;
    else N = 3 * N + 1;
}
System.out.println(N);
  
```

**inizializzazione indice del ciclo**

**Incremento dell'indice**

**Corpo del While**

**% rappresenta l'operatore di modulo**

**Sentinella. Il ciclo termina quando  $N$  è uguale al valore della sentinella (1).**

**// finché  $N$  è diverso da 1**

**// stampa  $N$**

**// se  $N$  è pari**

**// dividilo per 2**

**// altrimenti moltiplica  $N$  per 3 e sommi 1**

**// stampa  $N$**

# Flusso di Controllo: Do-While

Problema: quanti giorni saranno necessari per perdere metà del proprio peso se si dimagrisce del 2% ogni giorno?

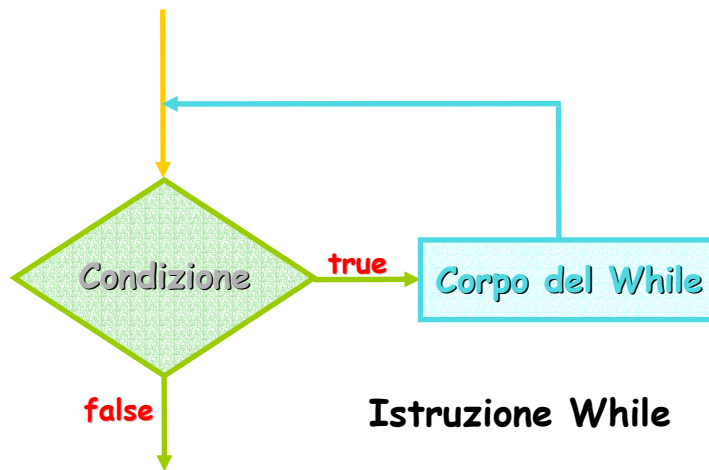
```
static public void main(String[] args) {
    ConsoleOutputManager video = new ConsoleOutputManager();
    double amtGrass = 100.0;           // inizializza il peso
    int nDays = 0;                     // inizializza il contatore dei giorni
    do {                                // continua ad
        amtGrass = amtGrass - amtGrass * 0.02; // · aggiornare il peso
        nDays = nDays + 1;             // · incrementare il n. di giorni
    } while (amtGrass > 50.0);         // finché non ha perso il 50% del peso
    return nDays ;                    // stampa il numero di giorni
} // main
```

Corpo del do-while

# Flusso di Controllo: While e Do-While

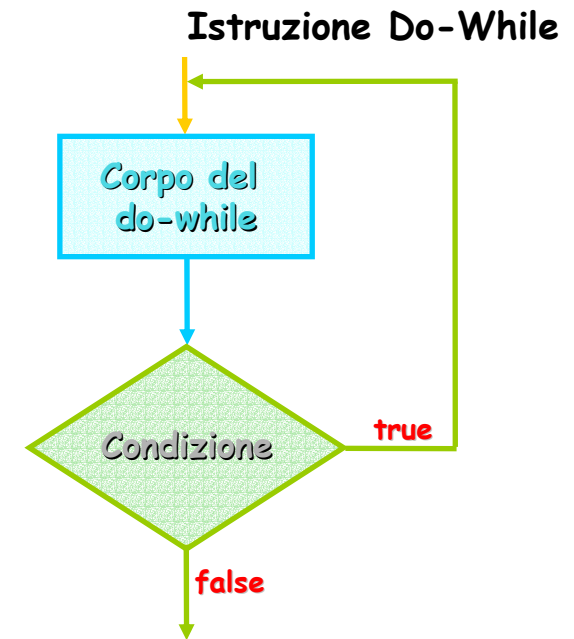
Al contrario dell'istruzione for, sia l'istruzione while che do-while non prevedono una sezione per l'inizializzazione e l'aggiornamento dell'indice del ciclo.

Semantica a confronto:



Sintassi Java:

```
while ( condizione ) corpo del while;
```



```
do  
    corpo del do-while  
while ( condizione );
```

# Es.: Calcolo della Media

**Problema:** Calcolare la media degli esami.

- I voti rappresentati come numeri interi saranno introdotti dalla tastiera. I dati termineranno quando si introdurrà il valore 9999.
- Verificare che il voto introdotto sia possibile (da 18 a 30), farlo reimmettere finché non sarà un voto possibile

**Suggerimenti:**

- Separare la routine di calcolo da quella di input
- Usare do-while per gestire la verifica/reimmissione dell'input.

# Es. : Calcolo della Media (Sol.)

```
import prog.io.*;

class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

# Es. : Calcolo della Media (Sol.)

Input/Output

```
import prog.io.*;
```

```
class MediaVoti {  
    static public void main(String[] args) {  
        ConsoleOutputManager video = new ConsoleOutputManager();  
        ConsoleInputManager tastiera = new ConsoleInputManager();  
        int esame, totale=0, i=0;  
        do {  
            i = i + 1; /* i++ */  
            do {  
                video.print("Immetti voto "+i+": ");  
                esame = tastiera.readInt();  
            } while (((esame<18) || (esame>30)) && (esame != 9999));  
            if (esame != 9999) totale = totale + esame;  
        } while(esame != 9999);  
        video.println("La media è: "+totale/(i-1));  
    }  
}
```

# Es. : Calcolo della Media (Sol.)

Controllo Se Voto Possibile

```
import prog.io.*;

class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

# Es.: Calcolo della Media (Sol.)

## Calcolo della Media:

- calcolo totale
- calcolo media

```
import prog.io.*;

class MediaVoti {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        ConsoleInputManager tastiera = new ConsoleInputManager();
        int esame, totale=0, i=0;
        do {
            i = i + 1; /* i++ */
            do {
                video.print("Immetti voto "+i+": ");
                esame = tastiera.readInt();
            } while (((esame<18) || (esame>30)) && (esame != 9999));
            if (esame != 9999) totale = totale + esame;
        } while(esame != 9999);
        video.println("La media è: "+totale/(i-1));
    }
}
```

abbreviazione

perché?

la media è intera

# Es.: Numeri Primi

**Problema:** Calcolare e stampare i primi 100 numeri primi.

- I possibili numeri primi sono generati automaticamente e non introdotti da tastiera.
- Sappiamo già che 1 e 2 sono primi quindi possiamo partire da 3 la ricerca.

**Suggerimenti:**

- Separare la verifica della primalità di un numero dal calcolo della sequenza di numeri primi.

# Es. Numeri Primi (Segue)

```
import prog.io.*;

class CalcoloPrimi {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        boolean èPrimo;
        int i=1, p=1, div;
        video.println("1 è primo!");
        do {
            p++;
            div = 2;
            èPrimo = true;
            while ((div <= Math.sqrt(p)) && èPrimo) {
                if (p%div == 0) èPrimo = false;
                else div++;
            }
            if (èPrimo) {
                i++;
                video.println(i+": "+p+" è primo!");
            }
        } while (i<100);
    }
}
```

# Es. Numeri Primi (Segue)

## Verifica di Primalità

```
import prog.io.*;

class CalcoloPrimi {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        boolean èPrimo;
        int i=1, p=1, div;
        video.println("1 è primo!");
        do {
            p++;
            div = 2;
            èPrimo = true;
            while ((div <= Math.sqrt(p)) && èPrimo) {
                if (p%div == 0) èPrimo = false;
                else div++;
            }
            if (èPrimo) {
                i++;
                video.println(i+": "+p+" è primo!");
            }
        } while (i<100);
    }
}
```

Radice Quadrata.  
Std Java library

Java è  
Unicode

// inizializzazione

// verifica primalità

// visualizza primi

# Es. Numeri Primi (Segue)

## Verifica di Primalità

```
import prog.io.*;

class CalcoloPrimi {
    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        boolean èPrimo;
        int i=1, p=1, div;
        video.println("1 è primo!");
        do {
            p++;
            div = 2;
            èPrimo = true;
            while ((div <= Math.sqrt(p)) && èPrimo) {
                if (p%div == 0) èPrimo = false;
                else div++;
            }
            if (èPrimo) {
                i++;
                video.println(i+": "+p+" è primo!");
            }
        } while (i<100);
    }
}
```

// p rappresenta la sequenza di possibili numeri primi  
// p è inizializzato a 1 (volendo potevo farlo partire da 2  
// p è incrementato ad ogni passaggio

// i conta quanti primi sono stati trovati  
// i è incrementato quando si trova un numero primo

// la ricerca si ferma quando i è 100

# Es. Numeri Primi (Esecuzione)

```
[16:10] cazzola@ulik:esercizi>java CalcoloPrimi
```

```
1 è primo!      21: 71 è primo!   41: 173 è primo!  61: 281 è primo!  81: 409 è primo!  
2: 2 è primo!   22: 73 è primo!   42: 179 è primo!  62: 283 è primo!  82: 419 è primo!  
3: 3 è primo!   23: 79 è primo!   43: 181 è primo!  63: 293 è primo!  83: 421 è primo!  
4: 5 è primo!   24: 83 è primo!   44: 191 è primo!  64: 307 è primo!  84: 431 è primo!  
5: 7 è primo!   25: 89 è primo!   45: 193 è primo!  65: 311 è primo!  85: 433 è primo!  
6: 11 è primo!  26: 97 è primo!   46: 197 è primo!  66: 313 è primo!  86: 439 è primo!  
7: 13 è primo!  27: 101 è primo!  47: 199 è primo!  67: 317 è primo!  87: 443 è primo!  
8: 17 è primo!  28: 103 è primo!  48: 211 è primo!  68: 331 è primo!  88: 449 è primo!  
9: 19 è primo!  29: 107 è primo!  49: 223 è primo!  69: 337 è primo!  89: 457 è primo!  
10: 23 è primo! 30: 109 è primo!  50: 227 è primo!  70: 347 è primo!  90: 461 è primo!  
11: 29 è primo! 31: 113 è primo!  51: 229 è primo!  71: 349 è primo!  91: 463 è primo!  
12: 31 è primo! 32: 127 è primo!  52: 233 è primo!  72: 353 è primo!  92: 467 è primo!  
13: 37 è primo! 33: 131 è primo!  53: 239 è primo!  73: 359 è primo!  93: 479 è primo!  
14: 41 è primo! 34: 137 è primo!  54: 241 è primo!  74: 367 è primo!  94: 487 è primo!  
15: 43 è primo! 35: 139 è primo!  55: 251 è primo!  75: 373 è primo!  95: 491 è primo!  
16: 47 è primo! 36: 149 è primo!  56: 257 è primo!  76: 379 è primo!  96: 499 è primo!  
17: 53 è primo! 37: 151 è primo!  57: 263 è primo!  77: 383 è primo!  97: 503 è primo!  
18: 59 è primo! 38: 157 è primo!  58: 269 è primo!  78: 389 è primo!  98: 509 è primo!  
19: 61 è primo! 39: 163 è primo!  59: 271 è primo!  79: 397 è primo!  99: 521 è primo!  
20: 67 è primo! 40: 167 è primo!  60: 277 è primo!  80: 401 è primo!  100: 523 è primo!
```

# Es.: Tavola di Verità

**Problema:** Calcolare tavole di verità.

- Calcolare la tavola di verità per:

$$x_1 \wedge (x_2 \vee \neg x_1) \wedge \neg x_3$$

- Visualizzare la tavola calcolata

**Suggerimenti:**

- Considerare falso uguale a 0, vero uguale a 1 e i connettivi \*, + e 1-x rispettivamente per  $\wedge$ ,  $\vee$ , e  $\neg$ .

# Es. Tavole di Verità

```
import prog.io.*;

class TavolaVerita { // 0 falso e 1 vero

    static public void main(String[] args) {
        ConsoleOutputManager video = new ConsoleOutputManager();
        int x1, x2, x3; // una variabile per ogni valore di verità

        video.println("Tavola di Verità per: f = x1 and (x2 or not x1) and not x3.\n");
        video.println("x1 x2 x3 | f\n-----|---");
        for (x1=0;x1<=1;x1++) // for annidati: uno per operando della formula
            for (x2=0;x2<=1;x2++)
                for (x3=0;x3<=1;x3++)
                    video.println(" "+x1+" "+x2+" "+x3+" | "+(x1*(x2+(1-x1))*(1-x3)));
    }
}
```

int perché bool  
è noncrescente

Il connettivo  
+ lavora in  $Z_2$

# Es. Tavole di Verità (Esecuzione)

```
[16:55]cazzola@ulik:esercizi>java TavolaVerita
```

```
Tavola di Verità per:  $f = x1 \text{ and } (x2 \text{ or not } x1) \text{ and not } x3.$ 
```

x1	x2	x3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

# Es.: Giochiamo a Carte 2

Riprendiamo l'esercizio visto nella 3<sup>a</sup> lezione.

Problema: scrivere un algoritmo che

- legga da tastiera due caratteri;
- i caratteri codificano le carte da gioco come segue:

A	asso	K	re
2-9	carte da 2 a 9	Q	quadri
0	dieci	C	cuori
J	fante	P	picche
Q	donna	F	fiori

- stampi la descrizione completa

Suggerimento: usare il costrutto switch.

# Es. Giochiamo a Carte 2

```
import prog.io.*;
```

```
class Queen {
```

```
    static public void main(String[] args) {
```

```
        ConsoleOutputManager video = new ConsoleOutputManager();
```

```
        ConsoleInputManager tastiera = new ConsoleInputManager();
```

```
        char seme, punto;
```

```
        punto = tastiera.readChar("punto: ");
```

```
        seme = tastiera.readChar("seme: ");
```

```
        switch (punto) {
```

```
            case '0': video.print("dieci"); break;
```

```
            case 'A': video.print("asso"); break;
```

```
            case '2': video.print("due"); break;
```

```
            ...
```

```
            case 'J': video.print("fante"); break;
```

```
            case 'Q': video.print("donna"); break;
```

```
            case 'K': video.print("re"); break;
```

conversione  
punto

conversione  
seme

e se tolgo  
il break?

```
        video.print(" di ");
```

```
        switch (seme) {
```

```
            case 'C': video.print("cuori");  
                break;
```

```
            case 'Q': video.print("quadri");  
                break;
```

```
            case 'F': video.print("fiori");  
                break;
```

```
            case 'P': video.print("picche");  
                break;
```

```
        }  
        video.println();  
    }
```

# Es. Giochiamo a Carte 2 (Esecuzione)

```
[19:10]cazzola@ulik:esercizi>java Queen
```

```
punto: A
```

```
seme: Q
```

```
asso di quadri
```

```
[19:10]cazzola@ulik:esercizi>java Queen
```

```
punto: J
```

```
seme: C
```

```
fante di cuori
```

```
[19:11]cazzola@ulik:esercizi>java Queen
```

```
punto: Q
```

```
seme: C
```

```
donna di cuori
```

# Esercizi

## Varianti:

- Risolvere l'esercizio sulla media degli esami calcolando la media pesata sui crediti, cioè somma dei voti moltiplicati per i crediti diviso il totale dei crediti;
- risolvere l'esercizio sui numeri primi usando il costrutto for anziché while;
- Modificare l'esercizio sulle tavole di verità in modo che funzioni anche nel caso di  $1 + 1 = 1$
- Modificare l'esercizio delle carte in modo che continui a chiedere delle carte finché non si introduce QQ.

## Esercizi:

- Calcolare  $a*b$  introdotti da tastiera usando solo gli operatori + e -. Suggerimento: ricordarsi come ci è stata insegnata la moltiplicazione alle elementari.