I cicli (loop)

Ciclo: gruppo di istruzioni che vengono ripetute

↓

corpo del ciclo ⇒ ogni sua ripetizione: iterazione del ciclo

1) istruzione **while**: ripete più volte una certa azione definita nel corpo finché un'espressione booleana di controllo rimane vera

⇓

Quando l'espressione booleana diventa falsa, la ripetizione termina

Istruzione che verifica, ad ogni iterazione, se una determinata espressione booleana è vera:

* se è **vera**: continua a ripetere questo blocco di istruzioni
* se è **falsa**: si interrompe la ripetizione

LISTATO 4.1 Un esempio di ciclo while.

```
import java.util.Scanner;

public class WhileDemo {
    public static void main(String[] args) {
        int conteggio, numero;

        System.out.println("Inserisci un numero");
        Scanner tastiera = new Scanner(System.in);
        numero = tastiera.nextInt();

        conteggio = 1; ⇒ inizializzazione var.
finché ← while (conteggio <= numero) {
            System.out.print(conteggio + ", ");
            conteggio++; ⇒ incremento var.
        }
        System.out.println();
        System.out.println("Sorpresa!");
    }
}
```

espressione booleana

corpo del ciclo: istruzioni ripetute

Esempio di output 2

```
Inserisci un numero
3
1, 2, 3,
Sorpresa!
```

Esempio di output 3

```
Inserisci un numero
0
```

← Il corpo del ciclo viene eseguito zero volte. ⇒ espressione booleana da subito falsa!

```
Sorpresa!
```

l'istruzione while potrebbe NON svolgere nessuna azione (se l'espressione booleana è inizialmente falsa)

Sintassi:

```
while (espressione_booleana){
    prima istruzione
    seconda istruzione
    ...
    ultima istruzione
}
```

2) istruzione **do-while**: simile al ciclo while, ma differisce perché il corpo di un ciclo do-while viene eseguito almeno 1 volta

Sintassi:

```
do{
    corpo del ciclo
} while (espressione_booleana);
```

"Fai questo blocco di istruzioni finché l'espressione booleana è vera"

* se è **vera**: torni su al do e ripeti le istruzioni
* se è **falsa**: prosegui eseguendo le istruzioni che seguono l'istruzione do-while

LISTATO 4.2 Esempio di ciclo do-while.

```
import java.util.Scanner;

public class DoWhileDemo {
    public static void main(String[] args) {
        int conteggio, numero;

        System.out.println("Inserisci un numero");
        Scanner tastiera = new Scanner(System.in);
        numero = tastiera.nextInt();

        conteggio = 1;
        do {
            System.out.print(conteggio + ", ");
            conteggio++;
        } while (conteggio <= numero);

        System.out.println();
        System.out.println("Sorpresa!");
    }
}
```

Esempio di output 1

```
Inserisci un numero
2
1, 2,
Sorpresa!
```

Esempio di output 2

```
Inserisci un numero
3
1, 2, 3,
Sorpresa!
```

Esempio di output 3

```
Inserisci un numero
0
1,
Sorpresa!
```

← Il corpo del ciclo viene sempre eseguito almeno una volta.

Il ciclo potrebbe NON interrompersi mai ⇒ Errore progettuale (ciclo infinito)

⇓

Le istruzioni presenti nel corpo alterano 1 o più variabili, in modo tale che l'espressione booleana di controllo diventi prima o poi falsa *

3) istruzione for:

Sintassi:

```
for (inizializzazione; espressione_booleana; aggiornamento){
    prima istruzione
    ...
    ultima istruzione
}
```

aggiorna la var. controllo dell'espressione booleana

es:

```
for(conteggio=1; conteggio<=3; conteggio++){
    corpo
}
```

inizializzazione ⇒ eseguita una sola volta (all'inizio)

NON va messo "e" ";" !!

⇓

Se si mette, si chiude il ciclo for senza alcuna istruzione (Non c'è il corpo)

LISTATO 4.5 Esempio di ciclo for.

```
public class ForDemo {

    public static void main(String[] args) {
        int contoAllaRovescia;

        for (contoAllaRovescia = 3; contoAllaRovescia >= 0; contoAllaRovescia--){
            System.out.println(contoAllaRovescia);
            System.out.println("attendere...");
        }

        System.out.println("Partito!");
    }
}
```

Esempio di output

```
3
attendere...
2
attendere...
1
attendere...
0
attendere...
Partito!
```

Di solito, all'interno di un'istruzione for, viene dichiarata una variabile

es:

```
int somma = 0;
for (int u = 1; u <= 10; u++)
    somma = somma + u * u;
```

var. locale al for → visibile solo all'interno del ciclo for

Programmare con i cicli:

- Cicli count-controlled: ciclo in cui si sa a priori per quante volte dovrò iterare

  ⇓

  ciclo for ⇒ + ADATTO!

es: quando voglio creare un programma che calcoli la media dei voti degli studenti (es. a pag. 151)

⇓

Quanto a priori è nr. degli studenti

- ask before iterating: se NON si conosce a priori il nr. di iterazioni da compiere, viene chiesto dell'utente se sia o no il momento di terminare l'iterazione

⇓

ciclo do-while ⇒ + ADATTO!

```
do {
    System.out.println("Scrivi sì o no se vuoi/non vuoi fare altri acquisti");
    risposta = tastiera.next();
} while (risposta.equalsIgnoreCase("si"));
```

⇓

condizione di terminazione del ciclo, se è FALSA!

se la risposta è sì ⇒ si ripete il ciclo

⇓

Adatto per certi tipi di INPUT

x luoghi classici di tipi di INPUT ⇒ valore sentinella

Valore sentinella: valore che, se immesso determina la fine dell'INPUT

⇓

viene stabilito in base ad indicazioni che a priori diamo dell'utente (in base alle condizioni imposte al programma)

Errori più comuni dei cicli:

- cicli infiniti indesiderati: blocco di istruzioni eseguito un nr. infinito di volte
- errori di one unit: il nr. di volte in cui viene eseguito il corpo del ciclo è sbagliato perché

eseguito una volta in più di quel che si desidera

eseguito una volta in meno di quel che si desidera

Relativo agli estremi