Costrutti base

- **IF**

*Il costrutto **if** si utilizza per dividere il codice in 2 blocchi di esecuzione di cui ne verrà eseguito solamente uno, in base alla veridicità (vero o falso) della condizione (a > 5) fra parentesi.*

```
if(condizione) {
    // operazioni
    // ...
} else {
    // operazioni
    // ...
}
```

Il costrutto ha alcune varianti (di cui seguono le immagini), per esempio:

- *se in un ramo c'è solo una istruzione (ovvero se serve solo 1 punto e virgola), si possono omettere le parentesi graffe (esempio 2), ma è una buona abitudine metterle sempre.*
- *si possono concatenare **else if** a piacere (esempio 3)*

2

```
if(a > 5)
    risultato = 10; // solo 1 operazione
else
    risultato = 15; // solo 1 operazione
```

3

```
if(a > 5) {
    // faccio robe
} else if(a == 5) {
    // faccio robe
} else if(a > 0) {
    // faccio robe
} else {
    // faccio robe
}
```

Si possono anche annidare più costrutti **if** uno dentro l'altro:

```
if(x > 0) {
    System.out.println("positivo!");
} else {
    if(x < 0) {
        System.out.println("negativo!");
    } else {
        System.out.println("zero!");
    }
}
```

Infine c'è anche una versione contratta di questo costrutto, di solito si utilizza per assegnare un valore diverso in base ad una condizione, ma è piuttosto complicato da leggere e quindi va usato solo se si è confidenti

```
b = (a > 5) ? 0 : 1;
```

Questo esempio qui si legge:
"se a > 5, assegna 0 a b, altrimenti assegna 1 a b"

- ***while***

```
while(condizione) {
    // operazione da ripete
    // operazione da ripetere
    // ...
}
```

Il costrutto **while** permette di **ripetere** un insieme di **operazioni** (quelle elencate fra le parentesi graffe che identificano il blocco del **while**) **finché una certa operazione rimane VERA**. Prima di iniziare ogni giro di questo ciclo, la condizione fra parentesi viene verificata, se è falsa il ciclo si interrompe e si esce dal **while**.

- **for**

```
for(inizializzazione; condizione_di_uscita; incremento) {
    // operazione da ripetere
    // operazione da ripetere
    // ..
}
```

Il costrutto **for**, come il while permette di **ripetere** un insieme di **operazioni** **finché una certa operazione rimane VERA**, ma fra parentesi ha due funzionalità in più:

1) permette di inizializzare un contatore di ciclo
2) permette di dichiarare già che tipo di operazione (di solito un incremento) effettuare al termine di ogni ciclo

```
for(i = 0; i < 10; i++) {
    // operazione da ripetere
    // operazione da ripetere
    // ..
}
```

Nell'immagine di sopra, si vede un **for** in cui si utilizza una variabile 'i' (che deve essere stata dichiarata per poter essere utilizzata) come contatore di ciclo che al primo giro viene inizializzata a 0. Il **for** "cicla" finché non diventa FALSO che i < 10, quindi in questo caso si uscirà dal for quando 'i' sarà esattamente uguale a 10, facendo un totale di 10 giri (per i valori di 'i': 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), perché alla fine di ogni ciclo, viene eseguita l'operazione i++ (incremento della variabile i di 1).

Segue un esempio di **for** dove si dichiara internamente la variabile 'i'

```
for(int i = 0; i < 10; i++) {
    // operazione da ripetere
    // operazione da ripetere
    // ..
}
```

In questo caso la variabile ‘i’ può essere utilizzata solamente in questo **for** (nelle parentesi graffe), e non esisterà più una volta terminato il **for** (ovvero una volta usciti dal blocco di istruzioni contenuto dalle graffe)

- **do while**

```
do {
    // operazione da ripetere
    // operazione da ripetere
    // ..
} while(condizione);
```

Il costrutto **do while** si comporta esattamente come un while, ma la condizione di uscita viene verificata alla fine di ogni giro e non all’inizio.
Di solito è un costrutto poco utilizzato.

- **switch**

Il costrutto **switch** serve nei casi in cui bisogna distinguere cosa fare in base ad un valore.
Si potrebbe utilizzare una sequenza di **if** per raggiungere lo stesso risultato, ma in base al quantità dei casi da riconoscere può diventare scomodo da implementare, quindi si opta per la sintassi più pulita dello **switch**.

Nell’immagine di esempio, se **n** vale 1 verrà eseguito l’insieme di operazioni fra “case 1:” ed il primo “break;”, se **n** vale 2 verrà invece eseguito l’insieme di istruzioni rispettivo al caso 2 e cosi via.

Il numero di casi da distinguere non è fisso e può essere modificato in base alle proprie esigenze.

Se **n** non assume nessuno dei valori elencati, viene eseguito il ramo default, se c’è (perché può essere omesso).

```
switch(n) {
    case 1:
        //operazioni
        // ...
        break;
    case 2:
        //operazioni
        // ...
        break;
    case 3:
        //operazioni
        // ...
        break;
    default:
        //operazioni
        // ...
        break;
}
```