I Cicli

In Java, un ciclo (o loop) è una struttura di controllo che consente di eseguire un blocco di codice più volte in base a una condizione. Utilizzare i cicli è fondamentale quando dobbiamo ripetere un'operazione più volte senza scrivere manualmente il codice per ogni singola esecuzione. Java offre tre principali tipi di cicli: for, while e do-while. Ognuno di questi ha caratteristiche specifiche e applicazioni ideali.

Ciclo for

Il ciclo for è il tipo di ciclo più comune quando si sa esattamente quante iterazioni devono essere fatte, ossia quando il numero di volte che il ciclo deve essere eseguito è noto in anticipo. Questo tipo di ciclo è utile, ad esempio, quando si ha a che fare con array o collezioni, dove si sa esattamente quante posizioni ci sono da iterare.

Sintassi del ciclo for

La sintassi di un ciclo for è la seguente:

```
for (inizializzazione; condizione; incremento) {
   // Codice da eseguire
}
```

- **Inizializzazione**: Viene eseguita una sola volta, prima che il ciclo inizi. Qui viene dichiarata e inizializzata la variabile di controllo del ciclo (ad esempio int i = 0).
- **Condizione**: Viene controllata prima di ogni iterazione del ciclo. Se la condizione è true, il corpo del ciclo viene eseguito; se è false, il ciclo si interrompe.
- **Incremento**: Questo è il passaggio che avviene alla fine di ogni iterazione, ed è usato per modificare la variabile di controllo, tipicamente incrementandola di 1 (ad esempio, i++).

Esempio di ciclo for

```
public class Main{
   public static void main(String[] args) {
      for (int i = 0; i < 10; i++) {
         System.out.println("Iterazione numero: " + i);
      }
   }
}
```

In questo esempio:

- **Inizializzazione**: int i = 0 – La variabile di controllo i è inizializzata a 0.
- **Condizione**: i < 10 – Il ciclo continua a iterare finché i è minore di 10.
- **Incremento**: i++ – Alla fine di ogni iterazione, i viene incrementato di 1.

Questo ciclo stampa i numeri da 0 a 9. La condizione viene verificata prima di ogni iterazione e il ciclo si interrompe quando i raggiunge il valore 10.

Variazioni del ciclo for

- **Ciclo for con step diversi**: Puoi anche specificare uno step di incremento diverso (ad esempio, i += 2 per incrementare di 2 invece di 1).
- **Ciclo for per decrementare**: Se vuoi decrementare la variabile di controllo, puoi usare un decremento (ad esempio i--).

Esempio di ciclo for con decremento:

```
public class Main{
   public static void main(String[] args) {
      for (int i = 10; i >= 0; i--) {
         System.out.println("Iterazione numero: " + i);
      }
   }
}
```

Ciclo while

Il ciclo while è utilizzato quando non si conosce il numero esatto di iterazioni che dovrà eseguire il ciclo, ma si sa quale condizione deve essere vera affinché il ciclo continui. Il ciclo while continua a eseguire il blocco di codice fino a quando la condizione che viene verificata all'inizio di ogni iterazione è vera.

Sintassi del ciclo while

La sintassi di base del ciclo while è la seguente:

```
while (condizione) {
   // Codice da eseguire
}
```

- **Condizione**: La condizione viene valutata prima di ogni iterazione. Se la condizione è vera, il ciclo continua. Se è falsa, il ciclo termina.

Esempio di ciclo while

```
public class Main{
   public static void main(String[] args) {
      int i = 0;
      while (i < 5) {
         System.out.println("Iterazione numero: " + i);
         i++;
      }
   }
}
```

In questo esempio:

- **Condizione**: i < 5 – Il ciclo continua finché i è minore di 5.
- **Incremento**: i++ – Alla fine di ogni iterazione, la variabile i viene incrementata di 1.

Questo ciclo stampa i numeri da 0 a 4.

Vantaggi del ciclo while

Il ciclo while è utile quando non si sa in anticipo quante iterazioni devono essere eseguite. La condizione che controlla l'uscita del ciclo può essere dinamica, ovvero dipendente da un input esterno o da una variabile che cambia durante l'esecuzione del ciclo.

Esempio di ciclo while con condizione dinamica

```
public class Main{
   public static void main(String[] args) {
      int numero = 0;
      while (numero != 5) {
         numero = (int) (Math.random() * 10); // Genera un numero casuale da 0 a 9
         System.out.println("Numero casuale generato: " + numero);
      }
      System.out.println("Il numero casuale generato è 5!");
   }
}
```

In questo caso, il ciclo continua a generare numeri casuali fino a quando non ne viene generato uno uguale a 5.

Ciclo do-while

Il ciclo do-while è molto simile al ciclo while, ma con una differenza fondamentale: nel ciclo do-while, il blocco di codice viene eseguito **almeno una volta**, anche se la condizione iniziale è falsa. Questo è perché la condizione viene verificata **dopo** l'esecuzione del ciclo.

Sintassi del ciclo do-while

```
do {
   // Codice da eseguire
} while (condizione);
```

- **Condizione**: Dopo ogni iterazione, la condizione viene verificata. Se è true, il ciclo continua. Se è false, il ciclo si interrompe.

Esempio di ciclo do-while

```
public class Main{
   public static void main(String[] args) {
      int i = 0;
      do {
         System.out.println("Iterazione numero: " + i);
         i++;
      } while (i < 5);
   }
}
```

In questo esempio, il ciclo esegue almeno una volta il blocco di codice anche se i fosse inizialmente maggiore di 5.

Vantaggi del ciclo do-while

Il ciclo do-while è utile quando si vuole eseguire un blocco di codice almeno una volta, indipendentemente dalla condizione iniziale. Un esempio pratico potrebbe essere un menu di opzioni che deve essere visualizzato almeno una volta all'utente, e poi continuare a essere mostrato fino a quando l'utente non seleziona una determinata opzione per uscire.

Parole chiave: break e continue

In Java, le parole chiave break e continue permettono di modificare il comportamento del ciclo in modo che:

- **break**: Interrompe completamente l'esecuzione del ciclo e salta il codice successivo.
- **continue**: Salta direttamente alla successiva iterazione del ciclo, saltando il resto del codice nel corpo del ciclo per quella iterazione.

Esempio di uso di break e continue

```
public class Main{
   public static void main(String[] args) {
      for (int i = 0; i < 10; i++) {
         if (i == 3) {
            continue; // Salta questa iterazione quando i è 3
         }
         if (i == 8) {
            break; // Interrompe il ciclo quando i è 8
         }
         System.out.println("Iterazione numero: " + i);
      }
   }
}
```

- Quando i è uguale a 3, la parola chiave continue salta quella iterazione, quindi non verrà stampato il numero 3.
- Quando i raggiunge il valore 8, la parola chiave break interrompe il ciclo.

In sintesi:

La struttura del ciclo for è ideale quando si conosce il numero di iterazioni, il ciclo while è più adatto quando non si conosce in anticipo il numero di iterazioni, e il ciclo do-while è utile quando si desidera che il ciclo venga eseguito almeno una volta.