Corso di base JAVA

Mauro Donadeo
mail: mauro.donadeo@gmail.com

Iterazioni

HT Lab

Complementi

Enunciato switch

Una sequenza di confronti in un'**unica variabile intera** con diverse **alternative costanti** può essere realizzata con l'enunciato switch

```
1  int x;
2  int y;
3  ...
4  if (x == 1)
5      y = 1;
6  else if (x == 2)
7      y = 4;
8  else if (x == 4)
9      y = 16;
10 else
11     y = 0;
```

```
1  int x;
2  int y;
3  ...
4  switch (x){
5      case 1: y = 1; break;
6      case 2: y = 4; break;
7      case 4: y = 16; break;
8      default: y = 0; break;
9  }
```

L'enunciato switch

- **Vantaggio:** non bisogna ripetere il nome della variabile;
- **Svantaggio:**
  - non si può usare se la variabile da confrontare non è intera
  - Non si può usare se uno dei valori da confrontare non è costante
  - in ogni `case` deve terminare con un enunciato `break`, altrimenti viene eseguito anche il corpo del `case` successivo. *Fonte di molti errori*

Errori con gli operatori relazionali

Alcune espressioni "naturali" con operatori relazionali sono errate, ma per fortuna il compilatore le rifiuta

- **if(0 <= x <= 1) NON FUNZIONA.**
- **if(0 <= x && x <= 1) OK**
- **if(x && y > 0) NON FUNZIONA**
- **if(x > 0 && y > 0) OK**

ESERCITAZIONE

Si scriva la classe **Triangolo** che descrive un triangolo, la cui interfaccia pubblica vi verrà mostrata tra poco. Collaudare la classe **Triangolo** con la classe di prova **TriangoloTester**, che legge i dati da standard input.

Problema

Riprendiamo un problema visto nella prima lezione, per il quale abbiamo individuato un algoritmo senza realizzarlo:

- **Problema:** Avendo depositato ventimila euro in un conto bancario che produce il 5% di interessi all'anno, capitalizzati annualmente, quanti anni occorrono affinché il saldo del conto arrivi al doppio della cifra iniziale?
- Abbiamo bisogno di un programma che ripeta degli enunciati (capitalizzazione degli interessi annuali, incremento del conto degli anni), finché non si realizza la condizione desiderata

Enunciato while

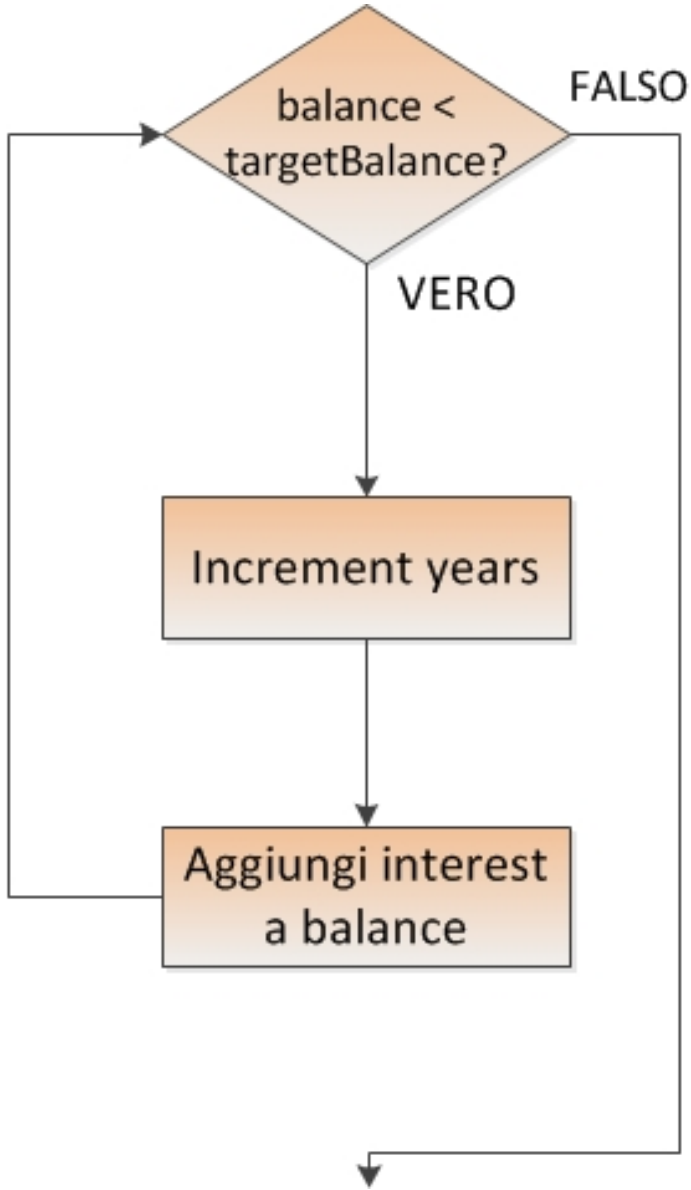
L'enunciato **while** consente la realizzazione di programmi che devono eseguire ripetutamente una serie di azioni finché è verificata una condizione

Algoritmo che risolve il problema

1. All'anno 0 il saldo è 20000
2. **Ripetere** i passi 2 e 3 **finché** il saldo è minore del doppio di 20000, poi passare al punto 4.
3. aggiungere 1 al valore anno corrente;
4. il nuovo saldo è il valore del saldo precedente moltiplicato per 1.05.
5. il risultato è il valore dell'anno corrente.

Il ciclo while

- Sintassi:
  - **while(condizione)**
    enunciato
- Scopo:
  - eseguire un enunciato finché la condizione è vera
- Il **corpo** del ciclo **while** può essere un enunciato qualsiasi, quindi anche un blocco di enunciati
- L'enunciato **while** realizza un **ciclo**



```
1  public class Investment{
2      public Investment(double aBalance, double aRate){
3          balance = aBalance;
4          rate = aRate;
5          years = 0;
6      }
7      public void waitForBalance(double targetBalance){
8          while (balance < targetBalance){
9              years++;
10             double interest = balance * rate / 100;
11             balance = balance + interest;
12         }
13     }
14     public double getBalance(){return balance;}
15     public int getYears(){return years;}
16     private double balance;
17     private double rate;
18     private int years;
19 }
```

```
1  public class InvestmentTester{
2      public static void main(String[] args){
3          final double INITIAL_BALANCE = 10000;
4          final double RATE = 5;
5          Investment invest = new Investment(
               INITIAL_BALANCE, RATE);
6          invest.waitForBalance(2 * INITIAL_BALANCE);
7          int years = invest.getYears();
8          System.out.println("The investment doubled after
               "+ years + " years");
9      }
10 }
```

Cicli infiniti

Esistono errori logici che impediscono la terminazione di un ciclo, generando un **ciclo infinito** L'esecuzione del programma continua **ininterrottamente**

```
1 int year = 0;
2 while (year < 20){
3     double interest = balance * rate / 100;
4     balance = balance + interest;
5 }
```

Cicli infiniti

Esistono errori logici che impediscono la terminazione di un ciclo, generando un **ciclo infinito** L'esecuzione del programma continua **ininterrottamente**

```
1 int year = 0;
2 while (year < 20){
3     double interest = balance * rate / 100;
4     balance = balance + interest;
5 }
```

```
1 int year = 20;
2 while (year > 0){
3     year++;
4     double interest = balance * rate / 100;
5     balance = balance + interest;
6 }
```

Ciclo for

For al posto di while

Molti cicli hanno questa forma:

i = inizio;
while(i < fine)
{*enunciati*; *i++*;}

Sostituzione

Per comodità esiste il ciclo for equivalente

for(i = inizio; i < fine; i++)

Non è necessario che l'incremento sia di una sola unità, né che sia positivo, né che sia intero.

for(inizializzazione; condizione; aggiornamento)
enunciato

Scopo

Eseguire un'**inizializzazione**, poi **ripetere** l'esecuzione di un enunciato ed effettuare un **aggiornamento** finché la **condizione** è vera;

Nota

L'inizializzazione può contenere la definizione di una variabile, che **sarà visibile soltanto all'interno del ciclo**

```
1 for (int y = 1; y <= 10; y++){
2     //corpo
3 }
4 //qui y non e' piu' definita.
```

Il metodo waitYears di Investment

Vogliamo sapere quale sarà il saldo finale del nostro investimento dopo 20 anni, con il tasso di interesse costante del 5%.

```
1  public class Investment{
2      ...
3      public void waitYears(int n){
4          for (int i = 1; i <= n; i++){
5              double interest = balance * rate / 100;
6              balance = balance + interest;
7          }
8          years = years +n;
9      }
10     ...
11 }
```

Visibilità delle variabili

Se il valore finale di una variabile di controllo del ciclo deve essere visibile al di fuori del corpo del ciclo bisogna definirla **prima** del ciclo.

Poiché una variabile definita nell'inizializzazione di un ciclo non è più definita dopo il corpo del ciclo, è possibile (e comodo) usare di nuovo lo **stesso nome in altri cicli**

```
1 double b = 0.3;
2 for (int i = 1; i < 10 && b < c; i++){
3     //modifica b
4 }
5 // qui b e' visibile, mentre i non lo e'
6 for (int i = 3; i > 0; i--)
7     System.out.println(i);
```

ESERCIZIO

Scrivere un programma che prende in ingresso una stringa e la inverte. Ad esempio se prende in ingresso **CIAO** dovrà stampare a video **OAIC**

Suggerimento

Il metodo `charAt(i)` restituisce il carattere della stringa alla i-esima posizione.

Cicli annidati

Problema

Vogliamo stampare una tabella con i valori delle potenze x^y, per ogni valore di **x** tra 1 e 4 e per ogni valore di **y** tra 1 e 5.

1	1	1	1	1
2	4	8	16	32
3	9	27	81	243
4	16	64	256	1024

Cicli annidati

Problema

Vogliamo stampare una tabella con i valori delle potenze x^y, per ogni valore di **x** tra 1 e 4 e per ogni valore di **y** tra 1 e 5.

1	1	1	1	1
2	4	8	16	32
3	9	27	81	243
4	16	64	256	1024

Innanzitutto stampiamo le prime 4 righe.

```
1 for (int x = 1; x <= 4; x++){
2     // stampa la riga x-esima della tabella
3 }
```

Per stampare la riga x-esima, bigogna stampare i valori x^1, x^2, x^3, x^5, cosa che si può fare facilmente con un ciclo **for**

```
1 for(int x = 1; x <= 4; x++){
2     //stampa la x-esima riga della tabella
3     for (int y = 1; y <= 5; y++){
4         int p = (int)Math.round(Math.pow(x, y));
5         System.out.print(p + " ");
6     }
7     System.out.println(); // va a capo
8 }
```

Come si può notare la soluzione del problema avviene mediante cicli **annidati**.

Il problema del ciclo e mezzo

Un ciclo del tipo:

- **fai qualocsa**, **Verifica** una condizione, **fai** qualcos'altro e ripeti il ciclo se la condizione è vera.

non ha una struttura predefinita in **Java** e deve essere realizzata con un trucco, come quello di usare una variabile **booleana**, detta **variabile di controllo** del ciclo.

Una struttura di questo tipo si chiama anche **ciclo e mezzo** o ciclo ridondante.

Situazione tipica: l'utente deve inserire un **insieme di valori**, la cui **dimensione non è predefinita**. Si realizza un ciclo **while**, dal quale si esce soltanto quando si verifica la condizione all'interno del ciclo.

```
1 boolean done = false;
2 while (!done){
3     System.out.println("Valore?");
4     String input = in.next();
5     if (input.equalsIgnoreCase("Q"))
6         done = true;
7     else
8         ... // elabora line
9 }
```

L

a lettera **Q** è un **valore sentinella** che segnala che l'immissione dei dati è terminata

Break, continue, e codice a spaghetti

Soluzione alternativa alla struttura "ciclo e mezzo"

- usare un ciclo infinito **while(true)** e l'enunciato **break**;
- L'enunciato **break** provoca la terminazione del ciclo;
- Esiste un unico enunciato **continue**, che fa proseguire l'esecuzione dalla fine dell'iterazione attuale del ciclo.

L'uso di **break** e **continue** è non necessario e **sconsigliabile**:

- perché contribuisce a creare **codice spaghetti**
- ovvero rappresentato da diagrammi di flusso pieni di linee, difficili da leggere e comprendere.

Esercitazione

1

Scrivere un programma che verifica se una stringa, introdotta dall'utente tramite lo standard input (un'intera riga), è una palindrome.

Suggerimento

Una stringa è una palindrome se è composta da una sequenza di caratteri (anche non alfabetici) che possa essere letta allo stesso identico modo anche al contrario (es. radar, anna, inni);

Esercitazione

2

Scrivere un programma che legga una sequenza di valori in virgola mobile inseriti dall'utente: il numero di valori non è predeterminato, dopo l'introduzione di ciascun valore l'utente deve premere il tasto "Invio" e la sequenza di valori termina non appena l'utente introduce una sequenza predeterminata di caratteri che non rappresenta un numero in virgola mobile. Dopo aver letto tutti i valori inseriti dall'utente, il programma ne deve visualizzare il valore medio e la deviazione standard.

Suggerimento

Per il calcolo della deviazione standard è consigliabile utilizzare la seguente formula: $D = \sqrt[2]{\frac{A - B * B/n}{n-1}}$ con n il numero di valori, B è la somma di tutti i valori, e la è la somma dei quadrati di tutti i valori.

Il ciclo do

Capita spesso di dover eseguire il corpo di un ciclo almeno una volta, per poi ripeterne l'esecuzione se è verificata una particolare condizione.

Esempio tipico: leggere un valore di ingresso, ed eventualmente rileggerlo finché non viene introdotto un valore valido.

- Si può usare un ciclo **while** innaturale

```
1 //si usa un'inizializzazione ingiustificata
2 double rate = 0;
3 while (rate <= 0){
4     System.out.println("Inserire il tasso:");
5     rate = console.readDouble();
6 }
```

- ma per comodità esiste il ciclo do

```
1 double rate;
2 do{
3     System.out.println("Inserire il tasso:");
4     rate = console.readDouble();
5 } while (rate <= 0);
```

Errori e consigli

Errori per uno scarto di uno

```
1 int years = 0; // o forse int years =1; ???
2 while (balance < 2 * initialBalance){
3     // o forse balance <= 2 * initialBalance ???
4     years++;
5     double interest = balance * rate / 100;
6     balance = balance + interest;
7 }
8 System.out.println("Investimento raddoppiato dopo " +
    years + " anni.");
```

- Provare con alcuni semplici casi di prova
- Investimento iniziale a 100 euro, tasso 50%
  - Allora years deve essere inizializzato a 0
- Tasso di interesse 100%
  - Allora la condizione deve essere $<$ e non $<=$