CORSO JAVA AESYS
Parte Prima

2022

www.aesystech.it

Classi ed Oggetti

- Le basi della programmazione object oriented
- Classi ed Oggetti
- I metodi in Java
- I metodi costruttori

Le basi della programmazione object oriented

Un linguaggio orientato agli oggetti è un linguaggio che supporta i concetti di classe ed oggetto. Questi due concetti sono strettamente legati tra loro. Se volessimo definire formalmente questi due concetti, dovremmo asserire:

- una **classe** è un'astrazione indicante un insieme di oggetti che condividono le stesse caratteristiche e le stesse funzionalità;
- un **oggetto** è un'istanza (ovvero una creazione fisica) di una classe.

Cerchiamo di capire meglio queste due definizioni. Ogni concetto che fa parte della teoria dell'Object Orientation infatti, esiste anche nel mondo reale. Questa teoria è nata proprio per soddisfare l'esigenza umana di interagire con il software in maniera più naturale. Si tratterà quindi solo di associare la giusta idea al nuovo termine. Passiamo subito ad un esempio pratico:

```
public class Punto
{
   public int x;     public int y;
}
```

Con Java è possibile creare codice per astrarre un qualsiasi concetto del mondo reale. Abbiamo appena definito una classe Punto. Evidentemente lo scopo di questa classe è definire il concetto di punto (a due dimensioni) tramite la definizione delle sue coordinate su di un piano cartesiano.

Classi ed Oggetti 1/3

Nel contesto della programmazione ad oggetti una classe dovrebbe limitarsi a definire che struttura avranno gli oggetti che da essa saranno istanziati. Istanziare, come abbiamo già affermato, è il termine object oriented che si usa in luogo di creare fisicamente, ed il risultato di un'istanziazione viene chiamato oggetto.

L'essere umano, per superare la complessità della realtà, raggruppa gli oggetti in classi. Per esempio, nella nostra mente esiste un modello definito dalla classe Persona anche se nella realtà esistono miliardi di oggetti di tipo Persona ognuno dei quali ha caratteristiche uniche.

Infatti, se volessimo trovare nel mondo reale un sinonimo di classe, dovremmo pensare a termini come idea, astrazione, concetto, modello o definizione. Quindi, definendo una classe Punto con codice Java abbiamo solo definito all'interno del nostro futuro programma il concetto di punto, secondo la nostra interpretazione e nel contesto in cui ci vogliamo calare (per esempio un programma di disegno). Con il codice scritto finora non abbiamo ancora definito nessun punto vero e proprio

Classi ed Oggetti 2/3

Quindi, come nel mondo reale, se non esistono punti concreti (ma solo la definizione di punto) non può accadere nulla che abbia a che fare con un punto. Ovvio anche che nel codice Java la classe Punto non dovrebbe essere eseguibile da sola! Occorre quindi definire gli oggetti punto, ovvero le creazioni fisiche realizzate a partire dalla definizione data dalla classe. Per esempio, istanzieremo oggetti dalla classe Punto creando un'altra classe che contiene un metodo main() che chiameremo TestOggettiPunto:

```
1   public class TestOggettiPunto
2   {
3   public static void main(String args[])
4   {
5   Punto punto1;
6   punto1 = new Punto();
7   punto1.x = 2;
8   punto1.y = 6;
9   Punto punto2 = new Punto();
10  punto2.x = 0;
11  punto2.y = 1;
12  System.out.println(punto1.x);
13  System.out.println(punto1.y);
14  System.out.println(punto2.x);
15  System.out.println(punto2.y);
16  }
17  }
```

Alla riga 5 troviamo la prima istruzione da eseguire: la dichiarazione di un oggetto di tipo Punto che viene chiamato punto1;
Alla riga 6 invece troviamo la sintassi che crea l'oggetto punto1 della classe Punto: Dalla riga 6 in poi è quindi possibile utilizzare l'oggetto punto1. Precisamente alle righe 7 e 8, assegniamo alle coordinate x e y del punto1 rispettivamente i valori interi 2 e 6; Notiamo l'utilizzo dell'operatore dot (che in inglese significa punto, ma nel senso del simbolo di punteggiatura ".") per accedere alle variabili x e y. L'istruzione punto1.x = 2, per esempio, assegna il valore intero 2 alla variabile x dell'oggetto punto1. Notare come le variabili x e y siano state dichiarate nella classe Punto, ma siano valorizzate per l'oggetto punto1, e quindi appartengono all'oggetto punto1.

Classi ed Oggetti 3/3

La classe Punto ci è servita solo come modello per istanziare punto1. Notiamo inoltre che all'interno della classe Punto le variabili x e y erano state dichiarate di tipo intero (int).
Ricapitolando, abbiamo prima dichiarato l'oggetto alla riga 5, l'abbiamo istanziato alla riga 6, e l'abbiamo utilizzato (impostando le sue variabili) alle righe 7 e 8.
Alla riga 9 poi, abbiamo dichiarato ed istanziato con un'unica riga di codice un altro oggetto dalla classe Punto, chiamandolo punto2:

```
Punto punto2 = new Punto();
```

Le definizioni di classi ed oggetto dovrebbero essere un po' più chiare: la classe è servita per definire come saranno fatti gli oggetti. L'oggetto rappresenta una realizzazione fisica della classe. In questo esempio abbiamo istanziato due oggetti diversi da una stessa classe. Entrambi questi oggetti sono punti, ma evidentemente sono punti diversi

L'operatore "." è sinonimo di "appartenenza". Sono quindi gli oggetti a possedere le variabili dichiarate nella classe (che chiameremo "variabili d'istanza", ovvero "variabili dell'oggetto"). Infatti, i due oggetti istanziati avevano valori diversi per x e y, il che significa che ognuno dei due oggetti punto1 e punto2 ha la sua variabile x e la sua variabile y. Le variabili di punto1 sono assolutamente indipendenti dalle variabili di punto2. Giacché le classi non hanno membri (variabili e metodi), non eseguono codice e non hanno un ruolo nell'ambito dell'esecuzione di un programma, e per quanto visto sono gli oggetti i protagonisti assoluti di quest'ambito. In pratica il rapporto tra classe ed oggetto è esattamente lo stesso che c'è tra idea e oggetto concreto nel mondo reale.

I metodi in Java

Nella definizione di classe, quando si parla di caratteristiche ci si riferisce ai dati (variabili e costanti) mentre col termine funzionalità ci si riferisce ai metodi. Infatti avevamo già accennato al fatto che il termine metodo è sinonimo di azione. Quindi, affinché un programma esegua qualche istruzione deve contenere metodi. Per esempio è il metodo main() che, per default, è il punto di partenza di ogni applicazione Java. Una classe senza metodo main(), come la classe Punto, non può essere mandata in esecuzione ma solo istanziata all'interno di un metodo di un'altra classe (nell'esempio precedente nel metodo main() della classe TestOggettiPunto). Il concetto di metodo è quindi anch'esso alla base della programmazione ad oggetti. Senza metodi, gli oggetti non potrebbero comunicare tra loro e i programmi non potrebbero eseguire azioni. Inoltre i metodi rendono i programmi più leggibili e di più facile manutenzione, lo sviluppo più veloce e stabile, evitano le duplicazioni e favoriscono il riuso del codice.

Dichiarazione di un metodo

[modificatori] tipo_di_ritorno nome_del_metodo ([parametri]) {corpo_ del_metodo}

- **Modificatori**
- **tipo di ritorno**
- **nome del metodo**
- **Parametri**
- **corpo del metodo**

I metodi costruttori

In Java esistono metodi speciali che hanno delle proprietà particolari. Tra questi c'è da considerare il metodo costruttore, che possiede le seguenti caratteristiche:

- ha lo stesso nome della classe;
- non ha tipo di ritorno;
- è chiamato automaticamente (e solamente) ogni volta che è istanziatoun oggetto, relativamente a quell'oggetto;
- è presente in ogni classe.
- Solitamente un metodo costruttore viene definito allo scopo di inizializzare le variabili d'istanza.

Quando creiamo un oggetto utilizzando la parola chiave new, c'è sempre una chiamata ad un costruttore (per esempio new Punto()). Eppure precedentemente non abbiamo mai fornito costruttori alle nostre classi, ma come appena detto, abbiamo chiamato costruttori ogni volta che abbiamo istanziato oggetti! Java infatti possiede una caratteristica molto importante che molti ignorano. Spesso il compilatore inserisce automaticamente e in modo trasparente istruzioni non inserite dal programmatore. Se infatti proviamo a compilare una classe sprovvista di costruttore, il compilatore ne fornirà uno implicitamente. Il costruttore inserito non contiene comandi che provocano qualche conseguenza visibile al programmatore. Esso è detto "costruttore di default" e non ha parametri.