Le interfacce e la clausola *implements*

L'interfaccia è un costrutto del linguaggio Java che permette di descrivere un insieme di *comportamenti* che dovranno essere necessariamente implementati dalle sue istanze.

La struttura base della dichiarazione di un'interfaccia è la seguente:

```
interface NomeInterfaccia
{
  // costanti
  // metodi astratti
}
```

La parola chiave **interface** serve per iniziare la dichiarazione di un'interfaccia ed è seguita da un nome che la identifica. Come per le classi, anche per le interfacce la convenzione impone di usare un nome con la lettera iniziale maiuscola. Tra le parentesi graffe si inserisce il contenuto dell'interfaccia, composto dalla dichiarazione delle costanti e dei metodi astratti, che stabiliscono il nome e la tipologia senza indicarne l'implementazione.

NOTA BENE

Le interfacce non possono contenere la dichiarazione di attributi.

ESEMPIO

Le interfacce sono usate per imporre la presenza di precisi metodi a tutte le classi che le implementano. La seguente interfaccia impone che tutte le classi implementino il metodo *getColore*.

```
interface OggettoColorato
{
  String CHIARO = "bianco";
  String SCURO = "nero";

  public String getColore();
}
```

Nell'interfaccia sono definite due costanti che sono automaticamente impostate dal compilatore come *static* e *final*. Per il metodo *getColore* non è prevista l'implementazione ma solo la dichiarazione con l'indicazione del livello di visibilità, del valore di ritorno e del nome.

Per istanziare un'interfaccia si deve prima dichiarare una classe che implementa l'interfaccia, usando la parola chiave **implements**, e successivamente istanziare questa classe.

```
class NomeClasse implements NomeInterfaccia
{
  // attributi
  // metodi
  // implementazione dei metodi dell'interfaccia
}
```

Una classe che implementa un'interfaccia può dichiarare attributi e metodi propri ma, in aggiunta, deve necessariamente fornire l'implementazione di tutti i metodi astratti ereditati dall'interfaccia.

ESEMPIO

La classe *Zaino* implementa l'interfaccia *OggettoColorato* fornendo il codice per il metodo astratto *getColore*.

```
class Zaino implements OggettoColorato
{
  private int volume;

  public Zaino(int vol)
  {
    volume = vol;
  }

  public String getColore()
  {
    if (volume < 30)
    {
      return "rosso";
    }
    else
    {
      return SCURO;
    }
  }
}
```

Si noti che se la classe *Zaino* non contenesse un'implementazione del metodo astratto *getColore*, il compilatore Java restituirebbe il messaggio di errore:

```
error: Zaino is not abstract and does not override abstract
method getColore() in OggettoColorato
```

Per provare il corretto funzionamento della classe *Zaino* si può eseguire il seguente programma.

```
class TestInterfaccia
{
  public static void main(String args[])
  {
    OggettoColorato oggetto;

    oggetto = new Zaino(20);
    System.out.println("Il colore è " + oggetto.getColore());
  }
}
```

Nel programma, il riferimento *oggetto* di tipo *OggettoColorato* è creato come istanza della classe *Zaino*.

L'implementazione di un'interfaccia è un concetto simile all'ereditarietà tra le classi, in comune hanno il meccanismo di ereditare i metodi. Nell'ereditarietà tra classi i metodi ereditati sono già implementati e possono essere sovrascritti con il meccanismo di *overriding*. Al contrario, nell'implementazione delle interfacce, i metodi ereditati devono essere implementati perché nella dichiarazione dell'interfaccia è stata definita solo la loro struttura. Un'ulteriore differenza, non prevista con l'ereditarietà tra classi, è la possibilità per una classe di implementare più di

una interfaccia, elencando i nomi delle interfacce dopo la clausola *implements* e separandoli con una virgola. Questo caso è indicato con il termine di **ereditarietà multipla** e prevede che la classe implementi i metodi di tutte le interfacce ereditate.

ESEMPIO

Nella programmazione guidata dagli eventi gli ascoltatori impongono, tramite un'interfaccia, i metodi che il programmatore dovrà implementare per gestire correttamente l'evento. L'interfaccia *ActionListener* dichiara il metodo *actionPerformed* per la gestione degli eventi legati alla pressione di un pulsante grafico. L'interfaccia *ListSelectionListener* dichiara il metodo *valueChanged* per la gestione degli eventi legati alla selezione dei valori da una lista grafica. Un gestore degli eventi che volesse gestire sia l'evento *pressione pulsante* che l'evento *selezione da lista* dovrebbe implementare entrambe le interfacce come mostra il seguente frammento di codice:

```
class GestoreEventi implements ActionListener,
ListSelectionListener
{
  public void actionPerformed(ActionEvent e)
  {
    // ...
  }

  public void valueChanged(ListSelectionEvent e)
  {
    // ...
  }
}
```