Operazioni sulle directory e sui file

La classe **File** nel package **java.io** permette di ottenere informazioni sui file, controllando l'esistenza, la data di ultima modifica oppure la presenza dell'attributo di sola lettura. Tramite la stessa classe si ha accesso anche ad un insieme di operazioni sulle directory, come la creazione o la cancellazione.

Il modo più semplice per creare un oggetto di classe *File* è indicare come parametro del costruttore il *pathname* del file o della directory a cui si vuole fare riferimento.

```
File documento = new File("c:\\dati\\registro.txt");
```

Si noti che, per indicare il separatore delle directory **** all'interno di una stringa, si è utilizzata la *sequenza di escape* ****

I metodi principali per la lettura delle informazioni sul file sono riassunti nella seguente tabella:

Metodo	Descrizione
exists()	Restituisce il valore *true* se il file esiste, altrimenti *false*.
isFile()	Restituisce il valore *true* se il file è un file, altrimenti *false*.
isDirectory()	Restituisce il valore *true* se il file è una directory, altrimenti *false*.
getName()	Restituisce una stringa con il nome del file o della directory.
getParent()	Restituisce una stringa con il nome della directory padre del file.
length()	Restituisce la dimensione del file in byte.
lastModified()	Restituisce il timestamp dell'ultima modifica al file.
canRead()	Restituisce il valore *true* se il file può essere letto, altrimenti *false*.
canWrite()	Restituisce il valore *true* se il file può essere scritto, altrimenti *false*.

I metodi principali per la modifica dei file sono riassunti nella seguente tabella:

Metodo	Descrizione
delete()	Cancella il file.
setReadOnly()	Imposta l'attributo di sola lettura al file.
renameTo(File)	Rinomina il file con il nome indicato nell'oggetto *File* passato come parametro.

L'oggetto di classe *File* può essere utilizzato anche per riferirsi alle directory. Per esempio:

```
File directory = new File("c:\\windows");
```

Con le directory, oltre ai metodi elencati precedentemente, si possono utilizzare anche i seguenti:

Metodo	Descrizione
list()	Restituisce un array di *String* con l'elenco dei file e delle sottodirectory contenuti nella directory.
listFile()	Restituisce un array di *File* con l'elenco dei file e delle sottodirectory contenuti nella directory.
mkdir()	Crea la directory il cui nome è stato indicato nel costruttore della classe *File*.

ESEMPIO

Realizzare un programma che mostra le informazioni su un file o il contenuto di una directory.

Il programma, per prima cosa, verifica l'esistenza del file o della directory. Successivamente, a seconda che si tratti di un file o di una directory, mostra le caratteristiche principali del file oppure l'elenco dei file contenuti nella directory.
La data di ultima modifica viene visualizzata trasformando il *timestamp*, restituito dal metodo *lastModified*, in un oggetto di classe *Date*, con l'istruzione:

```
new Date(doc.lastModified())
```

Per semplicità, il *pathname* del file è stato inserito direttamente nel codice del programma. Una estensione di questo progetto potrebbe prevedere la lettura del file da riga di comando. Il codice completo del programma è riportato di seguito.

PROGRAMMA JAVA (*ProgFile.java*)

```
import java.io.*;
import java.util.*;

class ProgFile
{
 public static void main(String args[])
 {
  File doc = new File("c:\\dati\\registro.txt");

  if (doc.exists())
  {
   if (doc.isFile())
   {
     System.out.println("Nome:       " + doc.getName());
     System.out.println("Directory:  " + doc.getParent());
     System.out.println("Dimensione: " + doc.length());
     System.out.println("Ult. mod. : " + new Date(doc.lastModified()));
     System.out.println("Sola lett.: " + !doc.canWrite());
   }

   if (doc.isDirectory())
   {
     File[] elenco = doc.listFiles();

     System.out.println("Elenco file");
     for(int i=0;i<elenco.length;i++)
     {
        System.out.println(elenco[i]);
     }
   }
  }
  else
  {
   System.out.println("File/directory inesistente.");
  }
 }
}
```