Lettura dei file di testo con FileInputManager (della libreria prog)

Dal capitolo 6 del libro Pighizzini e Ferrari vecchia edizione

Un file è un archivio di dati che si trova nella memoria di massa. Molte applicazioni utilizzate comunemente manipolano masse di dati organizzate in strutture dette basi di dati. Non tratteremo le organizzazioni degli archivi e delle basi di dati, ma ci limiteremo a presentare un particolare tipo di file impiegato per memorizzare testi. Questi file sono denominali file di testo. Ad esempio i testi sorgenti dei programmi vengono forniti al compilatore sotto forma di file di testo.

Un file di testo non è altro che un archivio organizzato come una sequenza di righe in cui ciascuna riga è una stringa, cioè una sequenza di caratteri.

Nel package prog. io vengono fornite due classi per la manipolazione dei file di testo: la

classe FilelnputManager e la classe FileOutputManager.

Per la lettura del file di lesto possiamo utilizzare la classe FilelnputManager, essa fornisce il seguente costruttore:

• public FilelnputManager(String nomefile)

Crea un canale di comunicazione per la lettura dal file il cui nome è specificato come

argomento. Se il file specificato non esiste, si verifica un errore in fase di esecuzione.

Per evitare che un'applicazione termini prematuramente in seguito al tentativo di costruire un

canale di comunicazione con un file che non esiste, è sempre bene verificare preliminarmente

resistenza del file. A tale scopo la classe FilelnputM anager mette a disposizione il metodo

statico:

• public static boolean exists(String nomeFile)

Restituisce true se esiste un file il cui nome è uguale alla stringa specificata come argomento, false in caso contrario.

Il principale metodo della classe è:

• public String readLine()

Legge una riga di testo dal file cui fa riferimento l'oggetto che esegue il metodo e la restituisce come risultato. Nel caso sia stata raggiunta la fine del file, restituisce null.

Un file di testo viene scandito sequenzialmente. Possiamo immaginare che vi sia un "puntatore al file" che indica la prossima riga a partire dalla quale cominciare a leggere. Al momento dell'"apertura" del file, cioè alla chiamata del costruttore, il puntatore è posto sulla prima riga del file. A ogni operazione di lettura, cioè a ogni chiamata di readLine, viene restituita la stringa su cui si trova il puntatore, che è spostato automaticamente alla riga successiva. Quando viene raggiunta la fine del file (end-of-file), la chiamata di readLine restituisce il riferimento null.

Esistono anche due metodi, reset e close, privi di argomenti e il cui tipo restituito è void: il primo riporta il puntatore all'inizio del file, il secondo chiude il canale di comunicazione.

Un’applicazione che elabori un file di testo può basarsi sullo schema seguente:

crea un canale di comunicazione per la lettura del file

esamina il file

chiudi il file

La fase di esame del file si incentrerà su un ciclo in cui si esamina una riga per volta. Il ciclo termina quando si raggiunge la fine del file, cioè quando la chiamata di readLine restituisce un riferimento null (si osservi che il file potrebbe anche essere vuoto, e dunque il codice interno al ciclo potrebbe essere eseguito zero volte).

Ecco una prima traccia di un metodo main che elabora un file di testo:

```
public static void main(String[] args) {
//legge il nome del file e crea l’oggetto corrispondente
String nomeFile = in.readLine("Nome del file da esaminare?");
//verifica l’esistenza del file
if (FileInputManager.exists(nomeFile)) {
  FileInputManager ingresso = new FileInputManager(nomeFile);
  ...
  String riga;
  while ((riga = ingresso.readLine()) f != null){
        ...elabora la riga...
  }
  //chiusura del file
  Ingresso.close() ;
  ...eventuali altre operazioni...
} else {
  ...comunica che il file non esiste...
}
```

In mancanza di una chiamata del metodo close, il canale di comunicazione tra un'applicazione e un file viene automaticamente chiuso al termine dell’esecuzione dell’applicazione.