Java

Stream e File

La classe File

- Per operare con l'intero file java mette a disposizione la classe File
- Per utilizzare la classe File è necessario importare la libreria **java.io.File**
- La classe File permette di operare con file su disco
- E' possibile recuperare informazioni sugli attributi del file.
- Le directory sono considerate particolari tipi di file.

Metodi principali

- canRead() ; Restituisce true se e' leggibile, altrimenti false
- canWrite() ; Restituisce true se e' scrivibile, altrimenti false
- equals(Object) ; Confronta il file con un altro
- exists() ; Restituisce true se il file esiste, altrimenti false.
- getPath() ; Restituisce il path relativo
- getAbsolutePath() ; Restituisce il path assoluto (es: c:\programmi\..\..)
- isDirectory() ; Ritorna true se esiste ed e' una directory.
- isFile() ; Ritorna true se esiste ed e' un file, altrimenti false.
- length() ; Restituisce la dimensione del file in byte
- list() ; Restituisce in un array di Stringhe i nomi dei file presenti in una directory.
- mkdir() ; Crea una dir e restitituisce true se ha avuto successo.
- renameTo(File) ; Rinomina un file e restitituisce true se ha successo.

Il concetto di flusso (stream)

- Un flusso (stream) è inteso come sequenza continua e monodirezionale di informazioni che transitano da un'entità a un'altra
- Un programma costituisce la sorgente o la destinazione di un flusso
  - L'altra estremità può essere un altro programma, un file su disco, lo schermo, la tastiera ...

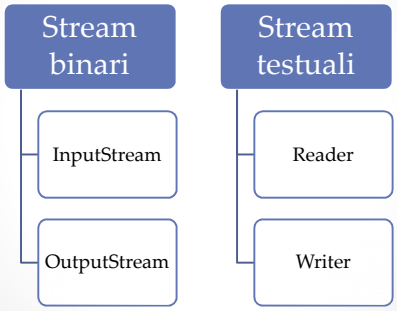
Stream: un esempio

- Nel caso di un programma che legge informazioni da un file su disco:
  - Il file costituisce la sorgente dello stream unidirezionale
  - Il programma costituisce la destinazione

Classi Java per input / output

- Le classi per input/output sono contenute nel package java.io
- Sono una gerarchia di classi organizzate in una struttura di ereditarietà in cui le sottoclassi estendono e specializzano le funzionalità base

Struttura delle classi



InputStream

- Classe astratta
- Opera su sequenze di byte
- Offre metodi per leggere i singoli byte
- Tutti i metodi possono lanciare IOException

InputStream - Metodi principali (1)

- int read()
  - Attende il prossimo byte, dopodiché ne restituisce il valore (0-255)
  - Restituisce -1 se il flusso è terminato
- int available()
  - Restituisce il numero di byte leggibili senza attesa

InputStream - Metodi principali (2)

- long skip(long n)
  - Salta i prossimi "n" byte dal flusso, se esistono
  - Ritorna il numero di byte scartati
- void close()
  - Chiude il flusso e rilascia le risorse di sistema associate

FileInputStream (sottoclasse di InputStream)

- Permette di leggere il contenuto di un file
  - Parametro del costruttore
  - Deve esistere ed essere leggibile
- Lettura sequenziale
  - Dall'inizio alla fine

FileInputStream (esempio)

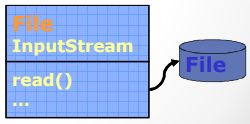
```
FileInputStream fileDaLeggere = null;
int valoreLetto;// leggo un int (1 byte sul file)
char c;      // carattere letto dal file
fileDaLeggere = new FileInputStream("prova.txt");
try {
  valoreLetto = fileDaLeggere.read();
   while (valoreLetto!=-1) {
    c = (char) valoreLetto;
    System.out.print(c);
    valoreLetto = fileDaLeggere.read();
  }
} catch (IOException e) {
    System.out.println("Errore: " + e + " nella lettura");
}
fileDaLeggere.close();
```

Classi filtro

- InputStream offre funzionalità minimali:
  - Permette solo di leggere byte
- Classi "filtro"
  - Arricchiscono le funzionalità o le prestazioni, interponendosi ad altre sorgenti o filtri
- Richiedono un InputStream da cui prelevare i dati
  - Deve essere passato nel costruttore
- Trasformano i dati letti da tale flusso
  - Conversione di formato, memoria tampone, reinserimento …

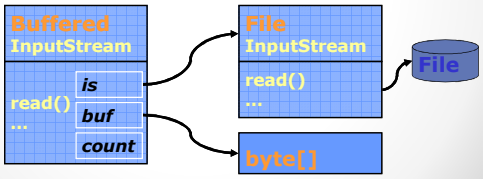
Classi filtro

```
BufferedInputStream bis;
bis=new BufferedInputStream(
        new FileInputStream("file.dat")
      );
byte b=bis.read(); b=bis.read();
```



Esempio BufferedInputStream

```
BufferedInputStream bis;
bis=new BufferedInputStream(
        new FileInputStream("file.dat")
      );
byte b=bis.read(); b=bis.read();
```



BufferedInputStream

- BufferedInputStream e BufferedOutputStream non offrono metodi differenti
- Migliorano l'efficienza bufferizzando gli accessi al file

DataInputStream

- DataInputStream e DataOutputStream forniscono metodi per la lettura di ogni tipo di dato
- int, double, String …

ObjectInputStream

- Con ObjectInputStream e ObjectOutputStream è possibile leggere e scrivere oggetti di classi serializzabili
- Una classe è serializzabile se implementa l'interfaccia Serializable
- Java.io.Serializable è un'interfaccia senza metodi

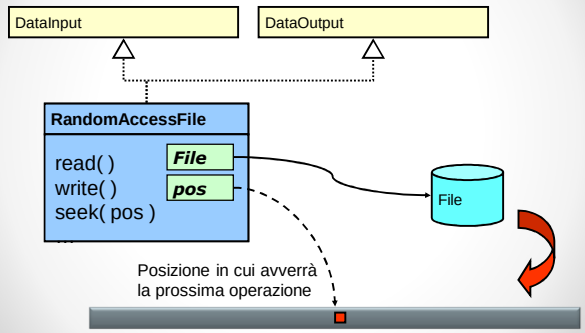
Reader Writer

- Classi per la lettura dei file di testo
- Usiamo per semplicità le classi che operano sui byte (derivate da InputStream OutputStream) anche per operare con i file di testo

File ad accesso casuale

- In alcune situazioni occorre operare su un file, procedendo in modo non sequenziale
  - Lettura e scrittura possono avvenire in qualsiasi posizione del file
  - indipendentemente dall'operazione precedente
- La classe RandomAccessFile modella il file come fosse un array di byte
  - Memorizzato su disco, invece che in memoria
  - Mantiene un puntatore di posizione interno che tiene traccia del prossimo byte a cui accedere
    - Modificabile dal programmatore
  - È possibile operare sia in lettura che in scrittura
    - Si specifica in fase di costruzione

File ad accesso casuale



Operare in lettura

- void seek(long pos)
  - Posiziona il puntatore interno a pos byte dall'inizio del file
- long getFilePointer( )
  - Restituisce la posizione corrente del puntatore interno rispetto all'inizio del file
- String readLine( )
  - Legge una sequenza di caratteri **ASCII** terminata da *newline* e la converte in formato Unicode
- String readUTF( )
  - Legge una sequenza di caratteri Unicode codificati nel formato UTF-8 (che contiene la lunghezza della stringa)
- Altri metodi dell'interfaccia DataInput
  - Permettono di leggere tipi elementari (numeri interi, numeri in virgola mobile, caratteri e booleani)

Operare in scrittura

- void writeBytes(String s)
  - Scrive la sequenza di byte meno significativi corrispondenti ai caratteri contenuti in "s"
- void writeChars(String s)
  - Scrive la sequenza di caratteri (due byte ciascuno) contenuti in "s"
- void writeUTF(String s)
  - Scrive la rappresentazione della stringa "s" nel formato UTF-8
- Altri metodi dell'interfaccia DataOutput
  - Permettono la scrittura di dati elementari