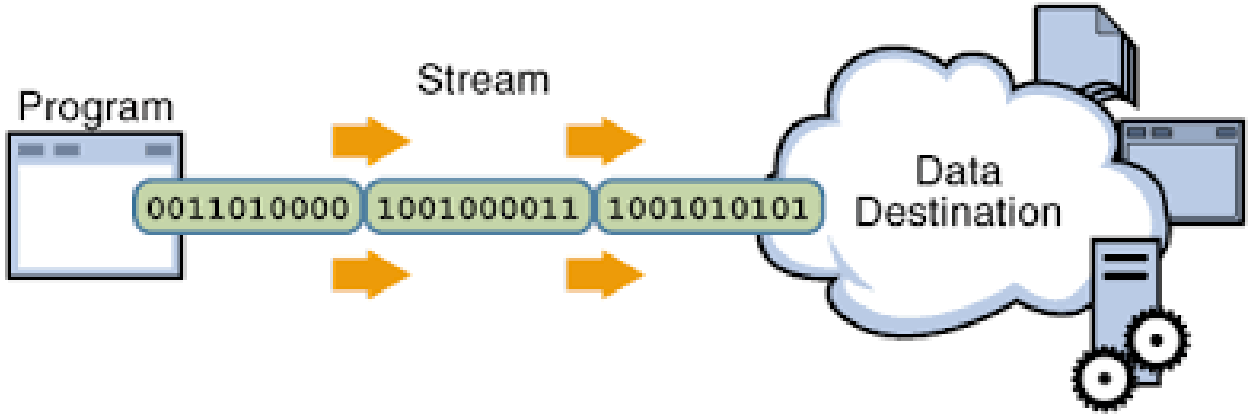


Stream e File

Java

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***
- è possibile recuperare informazioni sugli ***attributi*** del file
- le ***directory*** sono considerate particolari tipi di file

File
metodi principali (1)

- ***canRead***()
  - return true se il file è leggibile
- ***canWrite***()
  - return true se il file è scrivibile
- ***equals***(Object)
  - confronta il file con un altro
- ***exists***()
  - return true se il file esiste
- ***getPath***()
  - return il path relativo
- ***getAbsolutePath***()
  - return il path assoluto (es. c:\prog\..\..)

File
metodi principali (2)

- ***isDirectory***()
  - return true se esiste ed è una directory
- ***isFile***()
  - return true se esiste ed è un file
- ***length***()
  - return la dimensione del file in byte
- ***list***()
  - return in un array di Stringhe i nomi dei file presenti in una directory
- ***mkdir***()
  - crea una directory e restituisce true se ha avuto successo
- ***renameTo***(File)
  - rinomina un file e restituisce true se ha successo

File - esempio


```
File mioFile = new File("costituzione.txt");
if (!mioFile.exists()) {
  System.out.println("Il file " + mioFile + " non esiste!");
  return;
}
System.out.println("Informazioni sul File " + mioFile + ":\n");
String percorsoRel = mioFile.getPath();          // path relativo
String percorsoAss = mioFile.getAbsolutePath(); // path assoluto
System.out.println("Path relativo = " + percorsoRel);
System.out.println("Path assoluto = " + percorsoAss);
System.out.println("Lunghezza = " + mioFile.length() + " byte");
if(mioFile.isDirectory())
  System.out.println("File di tipo directory.");
if(mioFile.canRead())
  System.out.println("File leggibile.");
if(mioFile.canWrite())
  System.out.println("File scrivibile.");
}
```

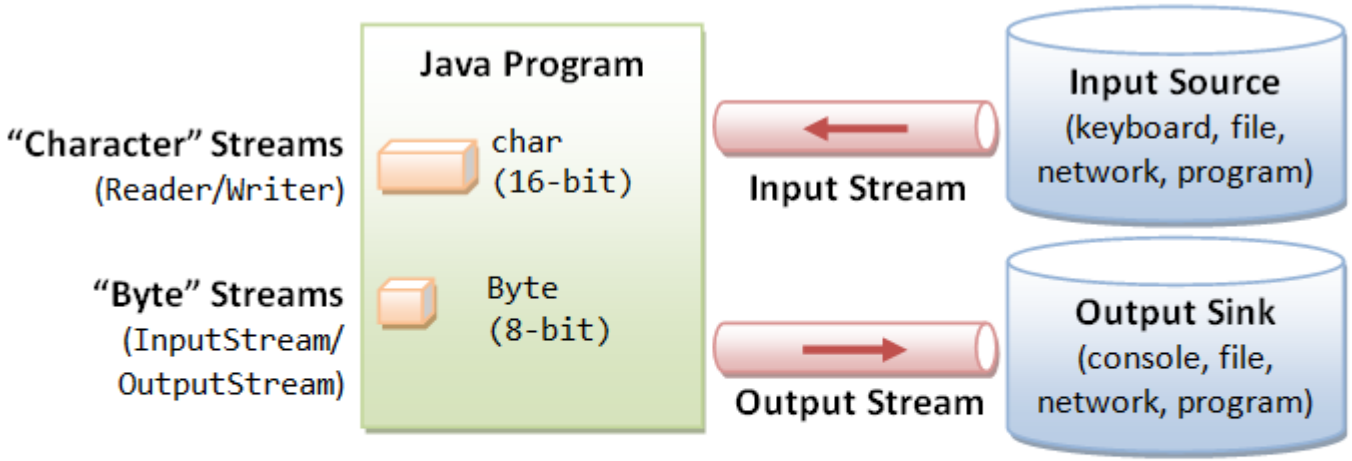
il concetto di flusso (stream)

- un ***flusso*** (***stream***) è una sequenza continua e ***monodirezionale*** di informazioni che transitano da un'entità a un'altra
- un ***programma*** può essere la ***sorgente*** o la ***destinazione*** di un flusso
- l'altra ***estremità*** può essere un altro programma, un ***file*** su disco, lo ***schermo***, la ***tastiera*** …
- esempio: ***programma*** che ***legge*** informazioni da un file su disco
  - il ***file*** è la ***sorgente*** dello stream unidirezionale
  - il ***programma*** è la ***destinazione***

classi Java per I/O

- 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

stream testuali e binari



Internal Data Formats:

- Text (char): UCS-2
- int, float, double, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

InputStream

- classe ***astratta***
- opera su ***sequenze di byte***
- offre metodi per ***leggere*** i ***singoli byte***
- tutti i metodi possono lanciare ***IOException***

InputStream metodi principali

- ***int read***()
  - attende il prossimo byte poi ne restituisce il valore (***0-255***)
  - return ***-1*** se il flusso è ***terminato***
- ***int available***()
  - return il ***numero di byte*** leggibili senza attesa
- ***long skip***(long ***n***)
  - ***salta*** i prossimi ***n byte*** dal flusso (*se esistono*)
  - return 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
  - il file deve ***esistere*** ed essere ***leggibile***
- lettura ***sequenziale***
  - dall'inizio alla fine

FileInputStream – esempio lettura carattere per carattere

A. Ferrari

```
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 dal flusso
  - ***conversione*** di formato, memoria buffer …

BufferedInputStream

- ***BufferedInputStream*** e ***BufferedOutputStream*** non offrono metodi differenti
- migliorano l'***efficienza bufferizzando*** gli accessi al file
- ***FileReader*** e ***BufferedReader*** per lettura da file di testo

BufferedReader – lettura riga


```
String nomeFile = "prova.txt";
try {
      String linea;
      File file = new File(nomeFile);
      FileReader fr = new FileReader(file);
      BufferedReader br = new BufferedReader(fr);

      while (( linea =  br.readLine()) != null){
        System.out.println(linea);
      }
      br.close();
   }
   catch(IOException e) {
     e.printStackTrace();
   }
```

DataInputStream

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

DataOutputStream - scrittura

A. Ferrari

```
DataOutputStream out = new DataOutputStream(
                                new FileOutputStream("ordini.dat"));
double[] prezzo = { 1.88, 1.75, 2.01 };
int[] quantita = { 83, 78, 97 };
String[] descrizione = { "Mele", "Pere", "Arance"};

for (int i = 0; i < prezzo.length; i ++) {
    out.writeDouble(prezzo[i]);
    out.writeInt(quantita[i]);
    out.writeUTF(descrizione[i]);
}
out.close();
```

DataOutputStream - lettura

A. Ferrari

```
DataInputStream in = new DataInputStream(new FileInputStream("ordini.dat"));
double prezzo;
int quantita;
String descrizione;
double totale = 0.0;
try {
   while (true) {
      prezzo = in.readDouble();
      quantita = in.readInt();
      descrizione = in.readUTF();
      System.out.println("ordine Kg." +
                         quantita + " " + descrizione + " Euro " + prezzo);
       totale += quantita * prezzo;
   }
}
catch (EOFException e) {
   System.out.println("Per un TOTALE di: Euro " + totale);
   in.close();
}
```

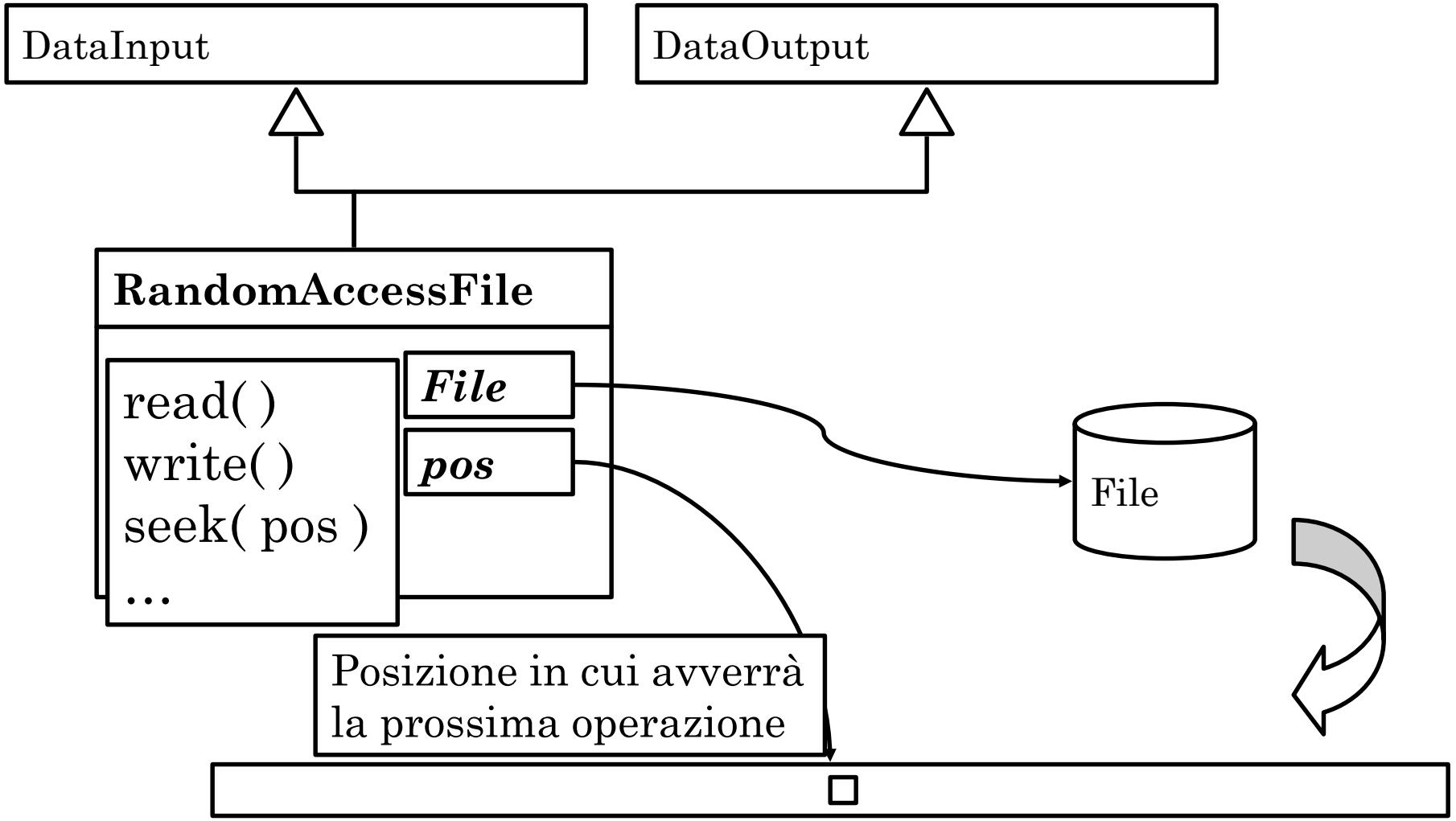
ObjectInputStream

- 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***

- classi per la lettura e scrittura 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

- permettono di operare su un file 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***


DataInput
DataOutput
RandomAccessFile
read( )
write( )
seek(pos)
…
File
pos
File
Posizione in cui avverrà
la prossima operazione

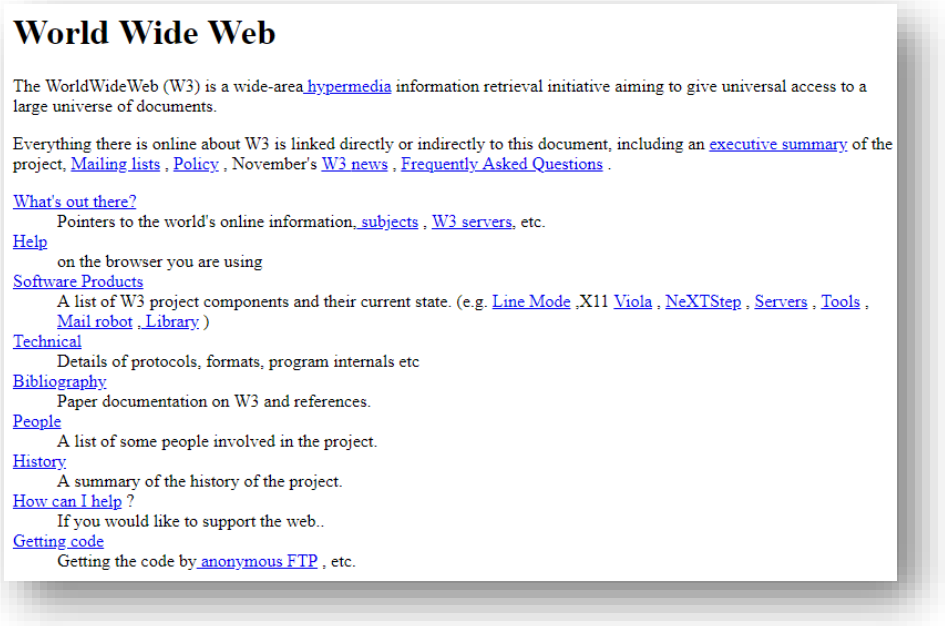
RandomAccessFile lettura

- void ***seek***(long ***pos***)
  - posiziona il ***puntatore*** a ***pos byte dall'inizio del file***
- long ***getFilePointer***( )
  - return la ***posizione corrente*** del puntatore 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)

RandomAccessFile scrittura

- void ***writeBytes***(String s)
  - scrive la ***sequenza di byte*** 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***

- uno stream può essere collegato a una sorgente dati diversa dal file
- nell'esempio lo stream in input fa riferimento a una pagina html che viene letta specificando l'URL



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

```
import java.io.*;
import java.net.*;

public class URLClient {
 public static void main(String[] args){
    try {
            URL url=new URL("http://info.cern.ch/hypertext/WWW/TheProject.html");
            URLConnection conn = url.openConnection();
            BufferedReader reader = new BufferedReader(
                        new InputStreamReader(conn.getInputStream()));
            String line = null;
            while ((line = reader.readLine()) != null)
                System.out.println (line);
            reader.close();
        }
        catch (MalformedURLException e) {
            System.out.println("MalformedURLException durante la connessione");
        }
        catch (IOException e) {
            System.out.println("IOException durante la connessione");
        }
    }
}
```