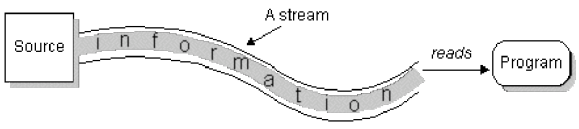
Lettura e scrittura – Stream

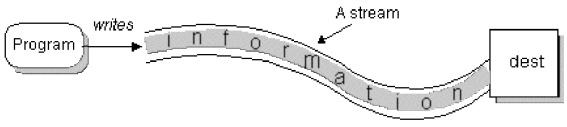
Obiettivo(fine della lezione): leggere/scrivere da/su file (e quindi verso qualsiasi dispositivo)

- Tipi di stream
  - Stream di caratteri
  - Stream di byte
- Classi astratte che implementano gli stream
  - Lettura: Reader e InputStream
  - Scrittura: Writer e OutputStream
- Sottoclassi specializzate
- Filtri
- Stream particolari
  - Standard I/O
  - File
  - Stream associati ai socket -> Programmazione di rete

Generalità

- Stream: canale di comunicazione tra un programma (Java) e una sorgente (destinazione) da cui importare (verso cui esportare) dati
- L'informazione viene letta (scritta) serialmente, con modalità FIFO





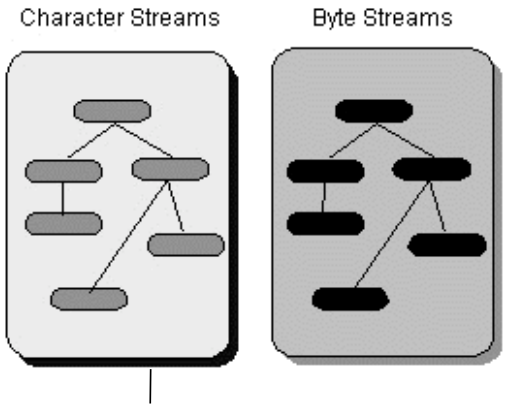
Generalità/2

- Accesso sequenziale in lettura e scrittura
- FIFO
- Read Only o Write Only: servono due stream per un canale bi-direzionale
- Bloccanti
- Quasi tutti i metodi possono generare eccezioni
- Il package java.io contiene classi che implementano gli stream

Lettura	Scrittura
Apri lo stream *while* (ci sono ancora dati) leggi dato chiudi lo stream	Apri lo stream *while* (ci sono ancora dati) scrivi dato chiudi lo stream

Tipi di Stream

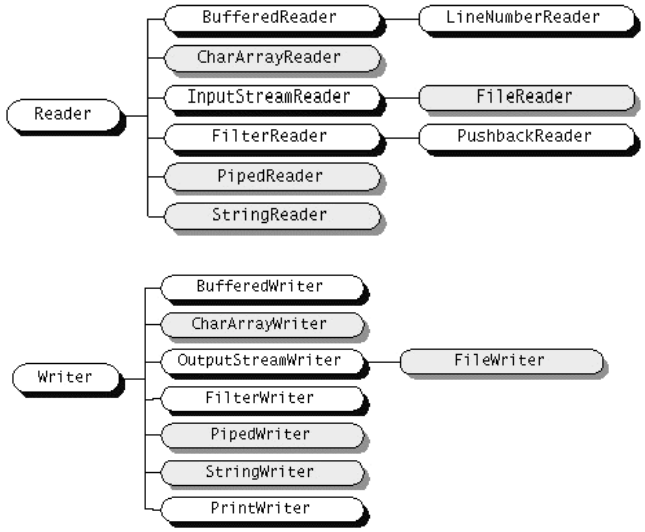
- Due gerarchie di classi:
  - Stream di caratteri: servono a leggere/scrivere sequenze di caratteri UNICODE a 16 bit
  - Stream di byte: servono a leggere/scrivere sequenze di byte (tipicamente dati)



Stream di caratteri

- Implementati dalle superclassi astratte Reader e Writer
- Reader: contiene una parziale implementazione e le API (metodi e campi) per realizzare stream che leggono caratteri
- Writer: contiene una parziale implementazione e le API (metodi e campi) per realizzare stream che scrivono caratteri
- Sottoclassi di Reader e Writer implementano stream specializzati (vedi prossima slide)

Stream di caratteri/2

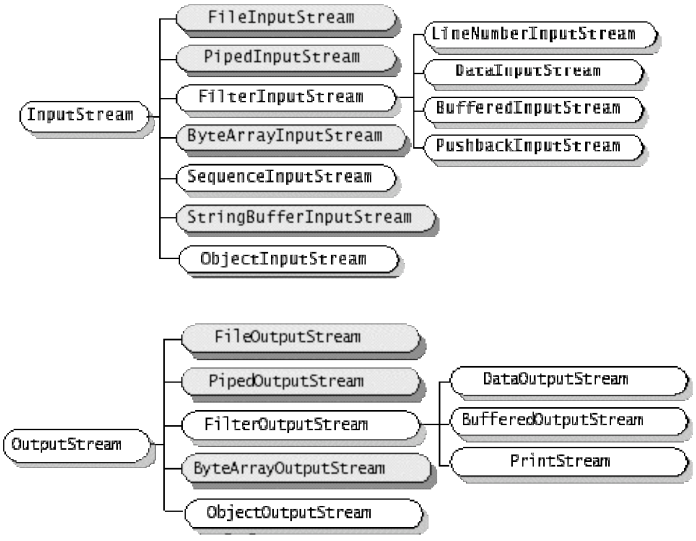


- Classi grigie: leggono/scrivono e basta
- Classi bianche: compiono anche qualche tipo di elaborazione

Stream di byte

- Implementati dalle superclassi astratte InputStream e OutputStream
- InputStream: contiene una parziale implementazione e le API (metodi e campi) per realizzare stream che leggono byte
- OutputStream: contiene una parziale implementazione e le API (metodi e campi) per realizzare stream che scrivono byte
- Sottoclassi di InputStream e OutputStream implementano stream specializzati (vedi prossima slide)

Stream di byte/2



- Classi grigie: leggono/scrivono e basta
- Classi bianche: compiono anche qualche tipo di elaborazione

Lettura – Classe Reader

- Costruttore: protected Reader();
- Metodi principali:
  - public int read() throws IOException:
    - Legge e restituisce il singolo carattere oppure –1 se arriva alla fine
  - public int read(char[] cbuf) throws IOException:
    - cbuf è il buffer di destinazione
    - Legge caratteri dallo stream in cbuf finché l'array è pieno, si verifica un errore o lo stream è finito
    - Restituisce no. caratteri letti o –1 se si raggiunge la fine dello stream
  - public abstract int read(char cbuf[], int off, int len) throws IOException:
    - cbuf è il buffer di destinazione
    - off è la posizione iniziale in cbuf a partire dalla quale scrivere
    - len è il numero max di caratteri da leggere
    - Restituisce il no. di caratteri letti o –1 se lo stream è finito
  - public abstract void close() throws IOException:
    - Chiude lo stream

Lettura – Classe InputStream

- Costruttore: public InputStream();
- Definisce gli stessi metodi, ma per leggere byte e array di byte:
  - public int read() throws IOException:
  - public int read(byte[] cbuf) throws IOException:
  - public abstract int read(char cbuf[], int off, int len) throws IOException:
  - public abstract void close() throws IOException:

Scrittura – Classe Writer

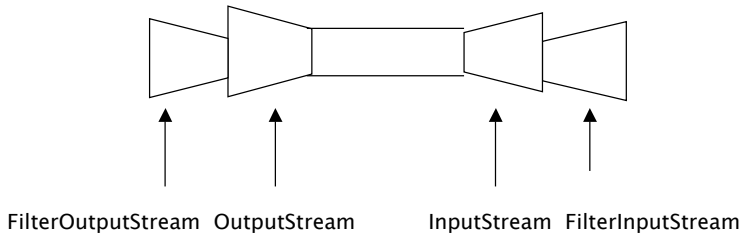
- Costruttore: protected Writer();
- Metodi principali:
  - public void **write**(int c) throws IOException
  - public void **write**(char[] cbuf) throws IOException
  - public abstract void **write**(char[] cbuf, int off, int len) throws IOException
    - cbuf: array di caratteri
    - off: posizione da cui cominciare a scrivere
    - len: No. caratteri da scrivere
  - public abstract void **flush**() throws IOException: svuota immediatamente il buffer (cbuf). Prosegui a cascata
  - public abstract void **close**() throws IOException: chiude lo stream

Scrittura – Classe OutputStream

- Costruttore: public OutputStream();
- Metodi principali:
  - public void **write**(byte[] b) throws IOException: scrive i b.length() byte di b sullo stream
  - public void **write**(byte[] b, int off, int len) throws IOException
    - b: array di byte
    - off: posizione da cui cominciare a scrivere
    - len: No. byte da scrivere
  - public abstract void **write**(int b) throws IOException
  - public abstract void **flush**() throws IOException: svuota immediatamente il buffer (b)
  - public abstract void **close**() throws IOException: chiude lo stream

Estensioni degli stream – Filtri

- Caratteristiche fondamentali dei filtri:
  - Sono classi derivate da InputStream e OutputStream
  - Si attaccano a un InputStream o OutputStream: chiamare write() su un filtro significa chiamare anche write() sull'OutputStream attaccato
  - E' possibile combinare in serie diversi filtri ottenendo combinazioni potenti
  - In pratica, si usano sottoclassi di FilterOutputStream e FilterInputStream



BufferedInputStream/ BufferedOutputStream

- Realizzano buffering per rendere più efficienti le operazioni di I/O su stream di byte
- I costruttori accettano un InputStream/OutputStream e realizzano un nuovo stream che fa I/O efficientemente
- I metodi sono sostanzialmente gli stessi (comunque controllare!!)
- Costruttori:
  - public BufferedInputStream(InputStream in); dimensione default buffer 512 byte
  - public BufferedInputStream(InputStream in,int size);
  - public BufferedOutputStream(OutputStream out); dimensione default buffer 512 byte
  - public BufferedOutputStream(OutputStream out,int size);
- BufferedOutputStream implementa flush()

DataInputStream/ DataOutputStream

- Forniscono metodi per leggere dati di formato complesso (byte, int, stringhe ecc.) da un InputStream
- Costruttori:
  - public DataInputStream(InputStream in)
  - public DataOutputStream(OutputStream out)

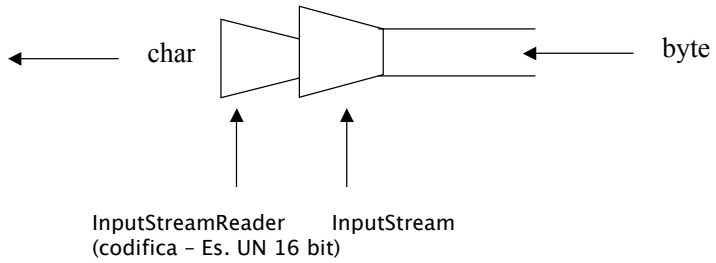
  Notare che InputStream e OutputStream sono astratte
- Metodi più importanti:
  - public writeBoolean(boolean v) throws IOException;
  - public boolean readBoolean() throws IOException;
  - .....
  - public writeChars(String s) throws IOException;
  - Per la lettura di stringhe è preferibile usare il metodo readLine() di BufferedReader -> più avanti

InputStreamReader/ InputStreamWriter

- Sottoclassi di Reader/Writer
- Creano un ponte tra stream a byte e a carattere
- Costruttori:
  - public **InputStreamReader**(InputStream in): accetta un oggetto InputStream e crea un oggetto InputStreamReader (e dunque Reader)
  - **OutputStreamWriter**(OutputStream out): accetta un oggetto di tipo OutputStream e crea un oggetto di tipo OutputStreamWriter (e dunque Writer)
  - Altri costruttori permettono di specificare una codifica

InputStreamReader/ InputStreamWriter/2

- La codifica permette di associare byte e caratteri
- Esempio: i byte dello stream sottostante possono rappresentare caratteri codificati in formato UNICODE a 16 bit
- In tal caso: le maiuscole vanno da u0041 a \u005a Hex



La situazione è analoga per un OutputStreamWriter

BufferedReader/ BufferedWriter

- Realizzano buffering per rendere più efficienti le operazioni di I/O su stream di caratteri
- Costruttori:
  - public BufferedReader(Reader in)
  - public BufferedReader(Reader in, int sz)
  - public BufferedWriter(Writer out)
  - public BufferedWriter(Writer out, int sz)

  sz permette di specificare la dimensione del buffer
- Metodo più importante:
  - public String **readLine**() throws IOException: legge una linea di testo, restituendo la stringa corrispondente

System.in/System.out

I/O standard

- in e out sono membri (variabili statiche final) della classe finale System
- in è di tipo InputStream, out di tipo OutputStream (è in realtà oggetto di classe PrintStream, una sottoclasse di OutputStream)

```
import java.io.*;
public class classeLettura {
    public static void main(String args[]) throws
    IOException {
            InputStream stdIn = System.in;
            String inputLine;

            InputStreamReader in = new
    InputStreamReader(stdIn);
            BufferedReader myReader = new
    BufferedReader(in);
            try {
                    inputLine = new
    String(myReader.readLine());
            }
            catch (IOException e) {throw e;}
            System.out.println("Hai scritto "+inputLine);
    }
}
```


FILE

- In Java sono rappresentati dalla classe FILE
- Rappresentazione astratta di file e directory
- Metodi per manipolare file e directory, ma non per leggere/scrivere
- Per leggere/scrivere su/da un file bisogna prima associare uno stream al file
- Nel seguito:
  - FileInputStream/FileOutputStream
  - Metodi utili della classe FILE
  - Problema: copia di file

FileInputStream/ FileOutputStream

- Sono sottoclassi di InputStream e OutputStream
- Aprono stream di byte da/verso file
- Hanno gli stessi metodi di InputStream e OutputStream
- Si possono applicare i filtri (ad esempio DataInputStream e DataOutputStream)
- Costruttori:
  - public FileInputStream(File file) throws FileNotFoundException
  - public FileInputStream(String name) throws FileNotFoundException

FileReader/ FileWriter

- Sono sottoclassi di Reader e Writer
- Aprono stream di caratteri da/verso file
- Hanno gli stessi metodi di Reader e Writer
- Si possono applicare i filtri (ad esempio BufferedReader e BufferedWriter)
- Costruttori:
  - public FileReader(File file) throws FileNotFoundException
  - public FileReader(String name) throws FileNotFoundException
  - public FileWriter(File file) throws FileNotFoundException
  - public FileWriter(String name) throws FileNotFoundException
  - public FileWriter(String name, boolean append) throws FileNotFoundException

I/O da FILE

- Scrivere una classe Java che apre un file testo "inputfile.txt" e lo copia su un file output.txt
  - Senza buffering
  - Con buffering
- Scrivere una classe Java che apre un file testo e ne stampa il contenuto, riga, per riga

La classe FILE

- Rappresentazione astratta di file e directory
- Metodi per manipolare file e directory (creare, distruggere ecc.)
- Non contiene metodi per leggere/scrivere
- Costruttori:
  - public File(String pathname) throws NullPointerException
  - public File(File parent, String child) throws NullPointerException
  - public File(String parent, String child) throws NullPointerException
- Se il file non esiste, esso è effettivamente creato solo quando si apre uno stream verso di esso

La classe FILE/2

- Campi più importanti:
  - public static final char separatorChar: restituisce il carattere (dipendente dal S. O.) che separa i nomi nelle directory
- Metodi più importanti:
  - public boolean delete() throws SecurityException
  - public boolean exists() throws SecurityException
  - public String getAbsolutePath()
  - public String getName()
  - public boolean isDirectory()
  - public boolean isFile()
  - public long length()
  - public String[] list()
  - public File[] listFiles()
  - public static File[] listRoots()
  - public boolean mkdir() throws SecurityException
  - public boolean mkdirs() throws SecurityException
- SecurityException è sottoclasse di RuntimeException

Esercizio

- Scrivere una classe Java con i seguenti metodi:
  1. public boolean creaDir(String nomeDir): crea (nella directory corrente) una sottodirectory avente per nome il contenuto di nomeDir. Restituisce false in caso di fallimento
  2. public File creaTextFile(String nomeFile, String parentDir): crea un file testo nella directory specificata in parentDir. Restituisce l'oggetto (il riferimento) corrispondente o null in caso di fallimento
  3. public boolean writeToFile(String text, File destFile): scrive il testo presente in text sul file specificato da destFile. Restituisce false in caso di fallimento
- Scrivere una programma Java che, dopo aver creato una directory "Prova", legga una stringa da stdin e la scriva su un file di nome "provafile.txt" nella directory "Prova"