

Programmazione 2

Esercitazione 10

Livio Pompianu - Simone Porcu

Cosa vedremo

- I/O streams
 - byte/character stream
 - buffered stream
 - I/O da linea di comando
- Try with resources

I/O streams

Un I/O stream rappresenta una sorgente di input (file, tastiera, url, ...) o una destinazione di output (file, memoria, altri programmi).



Fondamentalmente vi troverete a utilizzare due tipi di stream

- byte streams `InputStream` / `OutputStream`
- character streams `Reader` / `Writer`

Byte stream

Tutti i byte stream discendono dalle classi InputStream e OutputStream.

```
FileInputStream in = null;
```

```
FileOutputStream out = null;
```

```
try {
```

```
    in = new FileInputStream("xanadu.txt");
```

```
    out = new FileOutputStream("outagain.txt");
```

```
    int c;
```

```
    while ((c = in.read()) != -1)
```

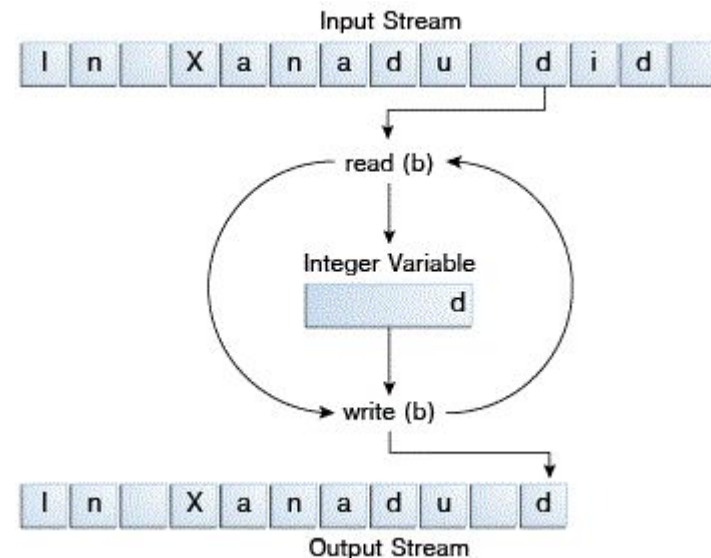
```
        out.write(c);
```

```
catch(...){ ... }
```

```
finally {
```

```
    if (in != null) {in.close();}
```

```
    if (out != null) {out.close();} }
```



Byte stream

Chiudere sempre gli stream!

Chiudere gli stream nel blocco **finally** garantisce che gli stream vengano chiusi anche in caso venga lanciata un'eccezione.

Quando non usare gli stream di byte

Le operazioni sui byte sono considerate operazioni a basso livello. Quando state operando su stream testuali è preferibile utilizzare stream di **caratteri**.

Perché conoscerli?

Perché tutti gli altri stream sono implementati a partire da uno stream di byte.

Character stream

Tutti i character stream discendono dalle classi Reader e Writer.

```
FileReader inputStream = null;
```

```
FileWriter outputStream = null;
```

```
try {  
    inputStream = new FileReader("xanadu.txt");  
    outputStream = new FileWriter("characteroutput.txt");  
    int c;  
    while ((c = inputStream.read()) != -1)  
        outputStream.write(c);  

```

```
} catch(...) { ... }
```

```
finally {  
    if (inputStream != null) {inputStream.close();}  
    if (outputStream != null) {outputStream.close();}  
}
```

Byte vs Character streams

I character stream **decodificano** il valore letto, restituendo un carattere (char).

I character stream sono spesso dei *wrapper* per i byte stream. Internamente usano un byte stream per le operazioni di I/O occupandosi solo della traduzione tra byte e caratteri.

`InputStream.read()` → restituisce un byte (8 bit) rappresentato come un `int`

`FileReader.read()` → restituisce un char (16 bit) rappresentato come un `int`

Buffered streams

```
BufferedReader inputStream = null;
PrintWriter outputStream = null;

try {
    inputStream = new BufferedReader(new FileReader("input.txt"));
    outputStream = new PrintWriter(new FileWriter("output.txt"));
    String l;

    while ((l = inputStream.readLine()) != null)
        outputStream.println(l);

} catch(...) { ... }

finally {
    if (inputStream != null) {inputStream.close();}
    if (outputStream != null) {outputStream.close();}
}
```

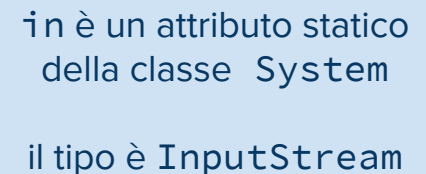

Scanning da tastiera

Oggetti del tipo `java.util.Scanner` dividono l'input in token (parti non divisibili) seguendo un determinato pattern.

Un token può essere **convertito** in un tipo particolare usando i diversi metodi `next` (`nextLong`, `nextInt`, ...).

Di default usa `Character.isWhitespace` per separare i token.
Viene spesso utilizzato per leggere input da tastiera.

```
Scanner sc = new Scanner(System.in);  
int i;  
  
i = sc.nextInt();    System.out.println(i);  
i = sc.nextInt(16); System.out.println(i);
```



`in` è un attributo statico
della classe `System`
il tipo è `InputStream`

Se non è in grado di convertire il valore letto, viene lanciata l'eccezione `InputMismatchException`

try with resources

```
BufferedReader br = new BufferedReader(new FileReader(path));  
try {  
    return br.readLine();  
}  
finally {  
    if (br != null) br.close();  
}
```

```
try (BufferedReader br =  
    new BufferedReader(new FileReader(path))) {  
    return br.readLine();  
}
```

All'interno del try è possibile usare solo le classi che implementano l'interfaccia `AutoCloseable` (che a sua volta estende `Closeable`).

Esercizi

Esercizio ContaParole

Scrivere una classe **ContaParole** che:

1. Acquisisce un testo da tastiera;
2. Suddivide il testo in singole parole;
3. Inserisce le parole in un insieme;
4. Stampa a video il numero di parole trovate.

Non è necessario avere un insieme ordinato. Si assuma che nel testo non ci possano essere parole ripetute.

Scegliere quale è la struttura più opportuna per gestire i dati tra quelle studiate.

Esercizio Stack

Creare una classe **Stack** che permetta la gestione di uno Stack di elementi generici, basandosi su un `ArrayList`. Si implementi:

- un costruttore;
- `push(elemento)`
- `pop`
- `size`
- `clear`

Esercizio Stack

Scrivere un programma che validi un insieme di parentesi letto in input da file.

Esempi di insiemi validati dal programma:

“()” “(())” “(()())”

Esempi di insiemi respinti dal programma:

“)(” “()” “((()” “(”

Esercizio Stack

Il programma:

- legge da **file.txt** in input le parentesi;
- usa lo stack per valutare il risultato (true o false)
- stampa il risultato su un file **result.txt**

Queries

Sia dato un array di stringhe con le ricerche effettuate dagli utenti su un motore di ricerca.

Calcolare quali sono le ricerche che rientrano nel $n\%$ delle richieste più popolari.

Esempio:

```
int n=30;
```

```
String[] queries = {"a", "g", "c", "d", "e", "g", "h", "a", "x", "y"};
```

Si ottiene come risultato:

"a", "g"

Queries

Per poter utilizzare l'algoritmo in un contesto reale, è necessario che sia in grado di gestire quantità enormi di dati in ingresso.

Scegliere opportunamente algoritmi e strutture dati da utilizzare in modo da ridurre sia la complessità in tempo che la complessità in spazio.

Scrivere un file di test per l'esercizio.

Esercizio Queries - File

Modificare l'esercizio Queries.

Il programma deve leggere i dati da un file che contiene, per ogni riga, una query differente. La lista di risultati deve essere salvata su file (sempre una query per riga).

La percentuale di query da analizzare, il nome del file di input ed il nome del file di output devono essere richiesti in input da tastiera.