

## Note ed esercizi aggiuntivi

### 15. Stream di I/O (conclusione)

*Esempio.* Conteggio del numero di righe e di lettere presenti in un testo inserito da tastiera. La fine del testo viene indicata mediante il segnale di *end-of-file*.

```
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;

class ContaLettere {

    public static void main(String[] args) throws IOException {
        //costruzione degli stream di caratteri
        InputStreamReader sr = new InputStreamReader(System.in);
        BufferedReader sorg = new BufferedReader(sr);
        System.out.println("Inserire il testo da esaminare:");

        int nRighe = 0;
        int nLettere = 0;

        String s;
        //lettura e conteggio
        while ((s = sorg.readLine()) != null) {
            nRighe++;
            for (int i = 0; i < s.length(); i++)
                if (Character.isLetter(s.charAt(i)))
                    nLettere++;
        }

        //chiusura dello stream
        sorg.close();
        sr.close();

        //comunicazione del risultato
        System.out.println("Il testo contiene " + nRighe +
            " righe e " + nLettere + " lettere");
    }
}
```

*Note*

Il riferimento predefinito `System.in` (campo statico di nome `in` della classe `java.lang.System`) è di tipo `InputStream` ed è normalmente associato alla tastiera (controllate la documentazione della classe `System`). In questo modo la tastiera è vista come una sorgente di byte. La classe `InputStreamReader` permette di vedere un flusso di byte come flusso di caratteri, fornendo un primo livello di astrazione, in cui i byte vengono raggruppati a due a due e interpretati come caratteri. In questo modo, il riferimento `sr` si riferisce a un flusso di caratteri che rappresenta la tastiera. L'utilizzo di `BufferedReader`, come negli esempi precedenti, fornisce un ulteriore livello di astrazione, strutturando sequenze di carattere in righe.

*Esempio.* Conteggio del numero di righe e di lettere presenti in un file di testo il cui nome viene specificato sulla riga di comando. Se non viene specificato alcun nome di file, il testo deve essere inserito da tastiera.

```
import java.io.InputStreamReader;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;

class ContaLettere {

    public static void main(String[] args) throws IOException {

        //costruzione degli stream di caratteri
        InputStreamReader sr;
        if (args.length == 0) {
            sr = new InputStreamReader(System.in);
            System.out.println("Inserire il testo da esaminare:");
        } else {
            sr = new FileReader(args[0]);
            System.out.println("Lettura dal file " + args[0]);
        }
        BufferedReader sorg = new BufferedReader(sr);

        int nRighe = 0;
        int nLettere = 0;

        String s;
        //lettura e conteggio
        while ((s = sorg.readLine()) != null) {
            nRighe++;
            for (int i = 0; i < s.length(); i++)
                if (Character.isLetter(s.charAt(i)))
                    nLettere++;
        }

        //chiusura dello stream
        sorg.close();
        sr.close();

        //comunicazione del risultato
        System.out.println("Il testo contiene " + nRighe +
            " righe e " + nLettere + " lettere");
    }
}
```

```
}  
}
```

#### *Note*

Si osservi l'utilizzo del tipo `InputStreamReader` (supertipo di `FileReader`) per la variabile `sr`.

#### **Esercizio 15.1**

Modificate l'applicazione dell'esercizio 14.3 in modo che, nel caso l'utente fornisca un solo argomento sulla linea di comando, visualizzi il file sorgente sul monitor. Se l'utente non fornisce alcun argomento l'applicazione deve copiare sul monitor le righe, man mano vengono inserite da tastiera (ispiratevi all'ultima versione di `VisualizzaFile`).

#### **Esercizio 15.2**

Modificate l'applicazione dell'esempio precedente in modo che tratti le situazioni di errore, intercettando opportunamente le eccezioni, anziché delegarle.

#### **Esercizio 15.3**

Modificate ulteriormente l'applicazione `CopiaFile` (esercizio 14.3) in modo tale che sia possibile utilizzarla con un comando nel seguente formato (le parentesi quadre indicano argomenti opzionali):

```
java CopiaFile [-v] [-c] <source_1> <source_2> ... <source_k> <dest>
```

Nell'invocazione base, l'applicazione deve copiare i contenuti dei file di testo di nome `<source_1>`, `<source_2>`, ..., `<source_k>` nel file `<dest>`.

Se viene indicata l'opzione `-c` (capitalize), durante la copia tutte le lettere minuscole devono essere convertite in maiuscole. Se viene indicata l'opzione `-v` (verbose) il risultato va visualizzato anche sul monitor. Ad esempio

```
java CopiaFile -c cane gatto topo
```

deve copiare nel file di nome `topo` il contenuto del file `cane` seguito da quello del file `gatto`, trasformando le minuscole in maiuscole. Se viene specificato un unico nome di file, l'applicazione deve leggere da tastiera e copiare in quel file. Se non viene specificato alcun nome di file, l'applicazione deve leggere da tastiera e inviare il risultato al monitor. In caso di errore l'esecuzione deve terminare fornendo un messaggio comprensibile all'utente.