FILE E FLUSSI

LEGGERE FILE DI TESTO

- Utilizzo package `java.io`
- Creo oggetto `FileReader`
    - apre in lettura il file
- Creo oggetto `Scanner`
- Collegare oggetto Scanner al FileReader

```
FileReader reader = new FileReader("input.txt");
Scanner in = new Scanner(reader);
String line = in.nextLine();
in.close();
```

Attenzione

- Ricorda di chiudere i file aperti in lettura `in.close()`
- `FileNotFoundException`: se il file non esiste
- importante chiudere reader: `reader.close()`
- È obbligatorio gestire `IOException`

RILEVARE LA FINE DELL'INPUT

- Basta usare `hasNext()` o `hasNextLine()`

SCRIVERE SU FILE DI TESTO

- Creo oggetto `PrintWriter` e collegarlo a un nuovo file
    - sovrascrive file precedente se presente

```
PrintWriter out = new PrintWriter("output.txt");
```

- Per abilitare append (e non sovrascrivere file)

```java
FileWriter writer = new FileWriter("file.txt", true);
PrintWriter out = new PrintWriter(writer);

out.println("Una frase");
```

Attenzione

- Ricorda di chiudere i file aperti in scrittura `out.close()`
- È obbligatorio gestire `IOException`

SCOMPOSIZIONE DI STRINGHE

- Utente inserisce + dati per riga
- Si può leggere l'intera riga (nextLine) e poi estrarre le sottostringhe relative ai singoli dati
  - ogni sotto-stringa delimitata da caratteri speciali (come spazi) è definita **token** o **lessem**

SOTTOSTRINGHE SEPARATE DA SPAZI

```java
String line = "Dato1 Dato2";
Scanner scan = new Scanner(line);

while (scan.hasNext()) {
    String token = scan.next();
    // elabora token
}
```

SOTTOSTRINGHE SEPARATE DA ALTRI DELIMITATORI

```java
Scanner scan = new Scanner("Dato1/Dato2/Dato3/Dato4");
scan.useDelimiter("/");

System.out.println("The delimiter in use is "+scan.delimiter());

while (scan.hasNext()) {
    String token = scan.next();
    // elabora token
}
```

REINDIRIZZAMENTO E CANALIZZAZIONE

- Re-indirizzamento dell'input standard
  - System.in sarà collegato al file

```
java IlMioProgramma < input.txt
```

- Re-indirizzamento dell'output standard
  - System.out sarà collegato al file

```
java IlMioProgramma > output.txt
```

- **Pipes**
  - se hai bisogno di passare output di un programma all'input di un altro
  - uso simbolo `|`

```
java IlMioProgramma < testo.txt | sort > testoOrdinato.txt
```

FLUSSO DI ERRORE STANDARD

- `System.err`
  - serve per comunicare degli errori all'utente