LABORATORIO DI INFORMATICA 1

2014-2015

- parte 6 -

Lettura da File

- Obiettivo: leggere un file
- Per prima cosa dobbiamo creare un oggetto che rappresenti il file che vogliamo utilizzare (inserendo anche l'estensione del file)
- Come nell'input da console utilizziamo uno **Scanner** per leggere i dati presenti nel file

```
File file = new File ( nomeFile );

Scanner scan = new Scanner ( file );
```

File

Per creare un File tramite la classe apposita, bisogna specificare il **percorso** (path) del file che vogliamo utilizzare.

- **Unix:** **/utente/unitn/informatica/laboratorio/prova.txt**
- **Windows:** **C:\unitn\informatica\laboratorio\prova.txt**

```
File file = new File ( “esempio.txt” );
```

Indicando solamente il nome del file, senza il percorso totale, il file “esempio.txt” deve trovarsi nella cartella principale del progetto.

Se indicate il percorso totale (assoluto) potete aprire qualsiasi file anche all’esterno della cartella del progetto

File

```
3     public static void main(String[] args) {
4         File file = new File(nomeFile);
5         Scanner scanner
6     }
7
8 }
9
```

- Import 'File' (java.io)
- Create class 'File'
- Change to 'FileLock' (java.nio.channels)
- Change to 'Files' (java.nio.file)
- Change to 'FileTime' (java.nio.file.attribute)
- Change to 'FileView' (javax.swing.filechooser)
- Rename in file (⌘2 R)
- Fix project setup...

Probabilmente ci verrà segnalato un errore. Basta cliccare sulla segnalazione sulla sinistra e importare le librerie necessarie (java.io per la classe File, java.util per la classe Scanner).

Questo è necessario perchè Java contiene moltissime classi e così specifichiamo cosa andiamo ad usare nel nostro programma

```
1  import java.io.File;
2  import java.util.Scanner;
3
4  public class InputScanner {
5
6      public static void main(String[] args) {
7          String nomeFile = "esempio.txt";
8          File file = new File(nomeFile);
9          Scanner scanner = new Scanner(file);
10     }
11
12 }
13
```

Add throws declaration
Surround with try/catch

Abbiamo un secondo errore, il programma ci chiede se:

generare un'eccezione (throw)

gestire l'errore (try - catch)

Questo avviene perchè non è detto che il nome del file che inseriamo sia corretto o che il file esista realmente.

```java
public static void main(String[] args) throws FileNotFoundException {
    String nomeFile = "esempio.txt";
    File file = new File(nomeFile);
    Scanner scanner = new Scanner(file);
}
```

Se decidiamo di generare un’eccezione (throw) vengono aggiunte queste due parole alla fine della dichiarazione del metodo. In questo caso viene generata un’eccezione nel caso in cui il file non venga trovato, che si chiama FileNotFoundException.

Ci sono vari tipi di eccezione che dipendono dall’errore che vanno a generare

```java
public static void main(String[] args) {
    String nomeFile = "esempio.txt";
    File file = new File(nomeFile);
    try {
        Scanner scanner = new Scanner(file);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```

Se decidiamo di gestire l'eccezione, avremo due nuovi blocchi di codice:

try -> qui dentro viene "provato" del codice se viene generato un errore il programma continua nel blocco **catch**

catch -> qui avviene la gestione effettiva dell'errore. In questo caso stampiamo solamente l'errore generato

Esercizio

1. Scrivere un programma che apra il file “esercizio1.txt” (scaricatelo da qui http://disi.unitn.it/~foroni/teaching/mat/esercizio1.txt e inseritelo nella cartella principale del progetto)

Il file contiene una serie di array di interi, la prima riga indica il numero di elementi per riga, mentre le righe successive sono effettivamente gli array. Stampate a video il massimo di ciascuna riga e alla fine il valore massimo totale.

Esempio:

```
5
1 2 9 2 3
4 9 1 5 6
0 6 51 2 6
12 54 2 7 4
1 4 5 90 4
```

Scrivere su File

- Per prima cosa dobbiamo creare un oggetto che rappresenti il file che vogliamo andare a scrivere (inserendo anche l'estensione del file)
- Utilizziamo una classe di Java che ci permette di scrivere su file: FileWriter, a cui passiamo in input il file che vogliamo scrivere
- Creiamo un "buffer" BufferedWriter che effettivamente scriverà sul file, a cui passiamo in input il FileWriter appena creato
- Per scrivere realmente utilizziamo il metodo write(quelloCheVogliamoScrivere) del buffer
- Quando abbiamo finito di scrivere, chiudiamo il nostro buffer con il metodo close ( )

Scrivere su File

```
String contenuto = “lezione di informatica”;

File file = new File ( nomeFile );

Scanner scan = new Scanner ( file );

FileWriter fileWriter = new FileWriter ( file );

BufferedWriter bw = new BufferedWriter ( fileWriter );

bw.write ( contenuto );     // scrivo la stringa contenuto (definita sopra) nel file

bw.close ( );
```

Esercizio 2

1. Scrivere un programma che apra il file “divina.txt” (scaricatelo da qui http://disi.unitn.it/~foroni/teaching/mat/divina.txt e inseritelo nella cartella principale del progetto)

Il file contiene il primo capitolo della divina commedia. Leggete linea per linea, contando il numero di a, e, i, o, u.

(utilizzate 5 variabili diverse o un array da 5 elementi, come preferite)

scorrete la stringa e controllate carattere per carattere con il metodo

nomeStringa.charAt (indiceDelCarattere)

Una volta letto tutto il file, create un nuovo file chiamato “risultati.txt” in cui scriverete il numero di a, e, i, o, u presenti