UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Fondamenti di Informatica (117IN)

A.A. 2022 / 2023

Lezione 13 - Java Files and Exceptions

Sylvio Barbon Junior
sylvio.barbonjunior@units.it

Sommario:

1) Lettura di File
    a) FileReader
    b) BufferedFileReader
2) Scrittura di File
    a) FileWriter
3) Eccezioni
    a) Gettare
    b) Catturare
    c) Propagare

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe File

- La classe File fornisce una rappresentazione astratta del percorso (pathname) di una cartella (directory) o di un documento (file);
- Sistemi operativi e interfacce utilizzano stringhe dipendenti dal sistema in uso. Questa classe mette a disposizione un'astrazione comunque gerarchica ma indipendente dal sistema;
- Esempi:
  - nei percorsi si usa il separatore "/" in Unix,
  - ma Windows ammette anche "\"

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe File

- Un percorso può essere **assoluto** o **relativo**. Quando è **assoluto**, contiene intrinsecamente tutte le informazioni per localizzare un file. Se invece è **relativo**, deve essere interpretato sulla base di altre informazioni;

- Di default, le classi nel package **java.io** risolvono i percorsi **relativi** rispetto alla directory corrente. Tipicamente, si tratta della cartella in cui si invoca la **Java Virtual Machine**

- Gli attributi statici **File.pathSeparator** e File.separator sono costanti che indicano i separatori usati nel sistema in uso (Es. ":" e "/")
  - pathSeparator è usato per separare vari percorsi in una lista (es. nella variabile di sistema PATH)
  - separator è usato per separare le varie parti che formano un percorso

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe File

- Costruttore principale: File(String pathname)
- boolean canExecute()
- boolean canRead()
- boolean canWrite() // testano i permessi da parte dell'applicazione
- boolean delete() // cancella il file o la cartella
- String getName() // restituisce il nome del file o directory
- String getParent() // restituisce il nome della cartella padre
- String getPath() // converte il nome astratto di un percorso a stringa
- long lastModified() // restituisce l'istante di ultima modifica
- long length() // restituisce la dimensione del file
- String[] list() // restituisce l'elenco di file e cartelle contenute nella cartella denotata dal path corrente (con numerose varianti…)

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe File

- Visto che la stessa classe può essere usata per effettuare operazioni su file e su directory, spesso occorre controllare il tipo di oggetto su cui si sta operando
- Il modo più semplice è invocare il metodo boolean isFile(), oppure boolean isDirectory()
- In aggiunta, il metodo boolean exists() verifica se il file o la directory esistono
- boolean isHidden() verifica se si tratta di file nascosto

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe FileReader

- FileReader è pensata per leggere dei file testuali
- Costruttori: FileReader(File file), FileReader(String fileName), …
- Metodi (è supportata la lettura sequenziale):
  - int read(): legge un carattere, restituito in uscita (-1 se si è giunti alla fine del file)
  - int read(char[] buffer, int offset, int length): legge length caratteri inserendoli alla posizione offset dell'array di caratteri buffer (restituisce il numero di caratteri effettivamente letti)
  - long skip(long n): avanza di n caratteri (restituisce l'avanzamento realmente effettuato)

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe FileReader

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

public class FileReaderExample {
    public static void main(String[] args) {
        try (FileReader reader = new FileReader("file_test.txt")) {
            int character;
            // Leggi il file carattere per carattere finché non arrivi alla fine
            while ((character = reader.read()) != -1) {
                // Stampa il carattere letto come un char
                System.out.print((char) character);
            }
        } catch (IOException e) {
            // Se si verifica un'eccezione durante la lettura del file, stampa il messaggio di errore
            System.err.println("Errore durante la lettura del file: " + e.getMessage());
        }
    }
}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe BufferedReader

- Un BufferedReader legge testo da un flusso di caratteri in ingresso, in modo da rendere efficiente la lettura sequenziale di caratteri e righe;
- Sarebbe possibile specificare anche la dimensione del buffer, ma quella di default è sufficientemente capiente per la maggior parte delle applicazioni;
- Tra i principali metodi:
  - int read() legge un singolo carattere,
  - String readLine() legge una riga di testo e
  - long skip(long n) salta nella lettura un certo numero di caratteri

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Classe BufferedReader

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

public class BufferedReaderExample {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("file_text.txt"))) {
            String line;
            // Leggi il file riga per riga finché non arrivi alla fine
            while ((line = reader.readLine()) != null) {
                // Stampa la riga letta
                System.out.println(line);
            }
        } catch (IOException e) {
            // Se si verifica un'eccezione durante la lettura del file, stampa il messaggio di errore
            System.err.println("Errore durante la lettura del file: " + e.getMessage());
        }
    }
}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Scrittura di File

- Si può scrivere su file usando le classi:
  - FileWriter,
  - BufferedWriter
- FileWriter scrive file di testo immediatamente su disco
- BufferedWriter scrive file di testo solo a buffer pieno, ottimizzando l'accesso al disco (in più offre il metodo newLine() per andare a capo)
- I costruttori/metodi sono simili a quelli visti in lettura (read > write)

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Eccezioni

- Quando si ha una situazione anomala in esecuzione…
  - divisione per 0,
  - accesso ad array con indice errato (ArrayIndexOutOfBoundException),
  - lettura oltre fine file, ecc.
- l'interprete Java: sospende l'esecuzione normale getta un'eccezione:
  - crea un'istanza di (una sottoclasse di) java.lang.Exception
  - (eventualmente) esegue codice per la gestione dell'eccezione
- Funzione principale:
  - Separare codice "normale" da situazioni anomale
  - Maggiore leggibilità

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Eccezioni (Gettare)

- Un'eccezione viene gettata
  - Durante l'esecuzione di un certo metodo, che avrà il suo record di attivazione sullo stack…
- Se viene catturata: viene gestita
- Altrimenti viene rimbalzata
  - al blocco più esterno (e al più esterno e…)
  - al chiamante (e al chiamante del chiamante…)
- Gettare = creare un'istanza di java.lang.Exception (o sottoclasse)
  - Chi lo fa:
    - L'interprete Java al verificarsi di una situazione anomala
    - Esplicitamente il programmatore Es.: throw new Exception();

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Eccezioni (Catturare)

- Costrutto try/catch/finally
- Esegue <I>
- Se <I> getta eccezione e, <I> si interrompe viene eseguita la prima <Ii> t.c. e instanceof <Ei>
- <id> è come un parametro formale <If> sempre eseguita alla fine

```
try { <I> }
catch (<E1> <id>) { <I1> }
catch (<E2> <id>) { <I2> }
catch (<En> <id>) { <In> }
finally { <If>}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Eccezioni (Propagare)

- Un'eccezione non catturata viene rimbalzata (propagata)
  - Al blocco esterno (ricorsivamente)
  - Al metodo chiamante (ricorsivamente)

- Se un metodo m() rimbalza eccezioni va dichiarato:
  - public void m() throws E1, E2 {
    - (se checked, ossia non RunTimeException)
    - (se unchecked, non serve)
- Regola generale: Meglio catturare che rimbalzare

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

```java
import java.io.*;

public class ProvaFile {
    public static void main (String[] args) throws IOException {
        File f = new File("file_test.txt");
        System.out.println(f.exists());
        f.createNewFile();
        System.out.println(f.exists());
        System.out.println(f.isFile());
        System.out.println(f.isDirectory());
        System.in.read();
        f.delete();
    }
}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Grazie!

Sylvio Barbon Junior - sylvio.barbonjunior@units.it