Directory comparator

Progetto d'esame per il corso di Programmazione a Oggetti

Prof. Giacomo Cabri

Traccia di progetto

REGOLE PER LO SVOLGIMENTO

Il progetto deve essere svolto usando il linguaggio Java e possedere i seguenti requisiti implementativi:

- Essere dotato di **interfaccia grafica** tramite cui interagire con il programma stesso.
- Sfruttare i meccanismi di **incapsulamento**, **ereditarietà** e **polimorfismo**.
- Per l'ereditarietà è possibile sfruttare **classi astratte** e **interfacce**; si considerano **escluse** le relazioni di ereditarietà diretta da classi di libreria Java.
- Sfruttare le classi di sistema Java per la gestione dell'**input/output**.
- Utilizzare almeno una **struttura dati** tra quelle presentate a lezione o derivate, oppure sfruttare i **generics** (a seconda di quanto fatto a lezione).
- Il programma deve essere eseguibile da **linea di comando**.

Il software deve essere accompagnato da pagine di **documentazione HTML** (ivi incluse le pagine generate tramite Javadoc) che descrivano le scelte di progetto effettuate e la struttura del sistema software.

Nel seguito del testo, i paragrafi evidenziati in azzurro sono **facoltativi**, e servono per differenziare il voto.

Lo svolgimento della parte **obbligatoria** contribuisce al voto per **25 punti**. Il contributo delle parti facoltative è riportato nelle rispettive descrizioni.

DESCRIZIONE DEL PROGETTO

Avere dischi rigidi sempre più capienti porta anche ad avere una enorme quantità di file, spesso duplicati e modificati in tempi diversi. Per gestire versioni diverse dello stesso file distribuite all'interno di un file system, esistono strumenti che permettono di confrontare il contenuto di due directory, riportando i file con lo stesso nome e confrontando sia la data di ultima modifica sia la dimensione.

Directory comparator

Si realizzi un software che implementi un comparatore di directory. Le funzionalità offerte dal programma si possono riassumere nei seguenti punti (dettagliati nei paragrafi successivi):

- Scelta delle directory da confrontare;
- Visualizzazione delle differenze in formato tabellare;
- Confronto ricorsivo;
- Ricerca di informazioni nel risultato;
- Stampa dei risultati;
- Aggiornamento delle directory.

Confronto e visualizzazione dei risultati

Come primo passo l'utente deve poter **selezionare le due directory** da confrontare. Fatto questo, il software **controlla** quali file sono presenti nelle due directory e ne confronta data, ora e dimensione.

Il **risultato** del confronto deve essere visualizzato in formato tabellare, elencando **tutti** i file presenti nelle due directory; per ogni file è necessario specificare:

- Se è presente in una sola directory o in entrambe;
- Nel caso in cui sia presente in entrambe, se la dimensione è uguale o diversa e se la data di ultima modifica (compresi ore, minuti e secondi) è uguale o diversa.

Si dia la possibilità di **salvare** i risultati ottenuti e di aprire risultati salvati in precedenza. Nel caso in cui si tenti di salvare dei risultati in un file che esiste già, deve essere chiesto all'utente se desidera **sovrascrivere** il file esistente [2 punti].

Si offra all'utente, inoltre, la possibilità di **ordinare** i risultati in base ai valori di una colonna [3 punti].

Confronto ricorsivo

Il programma deve permettere anche di eseguire un confronto **ricorsivo**. In particolare, in una directory possono essere presenti sia file sia sottodirectory. L'utente deve poter scegliere di effettuare anche il confronto tra sottodirectory che hanno lo stesso nome. Il confronto tra due sottodirectory implica il confronto tra i file che esse contengono.

Per implementare tale funzione si suggerisce di utilizzare il polimorfismo in Java.

Ricerca di informazioni nei risultati

L'utente deve avere la possibilità di effettuare delle **ricerche** nei risultati correnti. La ricerca si basa su **testo libero** che può essere una parte del testo contenuto in una cella della tabella dei risultati.

La ricerca deve evidenziare la **prima cella** che contiene il testo cercato; l'applicazione deve permettere all'utente di **continuare** la ricerca per evidenziare man mano le celle **successive** che rispondono ai requisiti (ad esempio, tramite un bottone "successivo").

Stampa dei risultati

Si dia all'utente la possibilità di **stampare** i risultati correnti. Si sfruttino le classi di libreria Java per stampare tramite una delle stampanti configurate dal sistema operativo [3 punti].

Aggiornamento delle directory

Si dia all'utente la possibilità di **aggiornare** le directory, copiando i file più recenti nella directory dove ci sono file più vecchi o mancanti [5 punti].

MATERIALE UTILE

- Classe `java.io.File` per la gestione file e directory in Java.
- Interfaccia Java `java.awt.print.Printable` e classe Java `java.awt.print.PrinterJob` per stampare su stampante. Si possono trovare tutorial online.