Università degli studi Insubria

“BOOK RECCOMENDER”

-

TECHNICAL MANUAL

Autori: UDDIN SHAKIM AHMED, KABUKA DAN MUMANGA, LANDINI MATTEO

Progetto Laboratorio A: BookRecommender

Versione: Settembre 2024

Sommario

1) Introduzione

BookRecommender è un progetto sviluppato nell'ambito del progetto di Laboratorio A per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria. Il progetto è stato realizzato utilizzando il linguaggio di programmazione Java e testato su sistemi operativi Windows 11.

Il progetto si articola in due componenti principali:

1. File di Codice: Contengono il codice Java che implementa le funzionalità dell'applicazione. Questo codice è responsabile delle operazioni che vengono mostrate all'utente, come la ricerca e visualizzazione di libri, la gestione di librerie personali, e l'interazione con il sistema di valutazioni e consigli.
2. Strutture Dati: Si tratta di file di testo che contengono informazioni persistenti come le credenziali degli utenti registrati, i dati relativi ai libri, le librerie create dagli utenti, le valutazioni dei libri e i consigli di lettura. Questi file supportano il funzionamento del sistema, memorizzando e recuperando le informazioni necessarie per l'applicazione.

1.1 Struttura del file Zip

Struttura della cartella compressa La cartella BookRecomender e sottocartelle necessarie per un'organizzazione ordinata dei file, di seguito le elenchiamo e ne spieghiamo il contenuto:

• bin: questa cartella i file necessari per l'esecuzione effettiva del programma, quindi il file .jar e quattro file di testo contenenti il Data Set iniziale (che per essere modificati ed acceduti devono risiedere nella stessa directory del programma .jar.

• doc: cartella nella quale risiede anche questo file, ossia la cartella che contiene la documentazione per la comprensione di tutto quello che riguarda il Book Reccomender, tra cui la Javadoc e il manuale utente.

• src: la cartella src contiene i file di codice sorgente denominati .java e .class che costituiscono il vero e proprio codice del programma.

• data: in questa cartella sono contenuti i file .txt usati dal programma come strutture dati per conservare credenziali degli utenti ed informazioni di sistema.

• autori.txt: semplicemente un file di testo contenente le informazioni degli autori del programma

•javadoc: è un sistema di documentazione usato in Java per creare automaticamente documenti HTML a partire dai commenti nel codice.

2) Struttura del programma

Il programma presenta diverse classi:

2.1 BookRecomnder:

main():

Il metodo main() (come dice il nome stesso) contiene tutte le chiamate agli altri metodi e rappresenta il metodo principale che viene eseguito. La struttura innestata del programma con chiamate a metodi esterni e la struttura ordinata del codice e resa possibile dalla presenza proprio del metodo main.

Il ciclo while (running) mantiene il programma in esecuzione fino a quando l'utente non sceglie di uscire (opzione 8).

cercaLibro()

Cerca un libro nel file Libri.dati in base alla query fornita (titolo, autore o autore + anno). Apre il file Libri.dati in modalità lettura e legge ogni riga del file verificando se contiene la query . Se trova una corrispondenza, stampa le informazioni del libro altrimenti stampa un messaggio che indica che nessun libro è stato trovato.

visualizzaLibro()

Visualizza i dettagli di un libro specifico e le sue valutazioni e consigli associati.

Apre il file Libri.dati e cerca una riga che inizia con il titolo se trova una corrispondenza, stampa i dettagli del libro e chiama il metodo mostraValutazioniEConsigli per visualizzare le valutazioni e i consigli del libro se non trova il libro, stampa un messaggio che indica che il libro non è stato trovato.

mostraValutazioniEConsigli()

Mostra le valutazioni e i consigli di lettura per un libro specifico utilizzando il file ValutazioneLibri.dati per le valutazioni e il file ConsigliLibri.dati per visualizzare i consigli.

registrazione()

Permette a un nuovo utente di registrarsi creando un nuovo record nel file UtentiRegistrati.dati. Richiede all'utente di inserire nome e cognome, codice fiscale, email, userID, e password successivamente crea un nuovo oggetto Utente con le informazioni inserite.

login()

Permette a un utente registrato di effettuare il login confrontando le credenziali inserite con quelle memorizzate nel file UtentiRegistrati.dati. Richiede all'utente di inserire userID e

password poi legge il file UtentiRegistrati.dati per verificare se le credenziali corrispondono a un utente registrato.

registraLibreria()

Permette all'utente autenticato di creare una nuova libreria e aggiungere libri ad essa, gli richiede di inserire un nome personalizzato per la libreria successivamente l'utente potrà inserire libri finche non digiterà la parola fine.

inserisciValutazioneLibro()

Permette all'utente autenticato di aggiungere una valutazione per un libro specifico gli richiede di inserire il titolo del libro e le valutazioni per vari aspetti (stile, contenuto, gradevolezza, originalità, edizione), analizza i dati e calcola il voto finale facendo la media tra tutti i dati.

inserisciSuggerimentoLibro()

Permette all'utente autenticato di aggiungere consigli di lettura per un libro specifico e gli permette di aggiungere fino a 3 libri consigliati, finché non digita "fine".

2.2 Consiglio:

Consiglio()

Questo è il costruttore della classe Consiglio. Viene utilizzato per creare un'istanza della classe Consiglio con l'ID dell'utente e il titolo del libro di riferimento. Inizializza anche la lista dei consigliati come una nuova ArrayList.

inserisciSuggerimentoLibro()

Aggiunge un libro alla lista dei consigliati, a condizione che il numero totale di libri consigliati non superi il limite massimo di tre. Se la lista contiene già tre libri, il metodo stampa un messaggio che indica che non è possibile aggiungere ulteriori consigli.

2.3 Librerie:

Libreria()

Inizializza un nuovo oggetto Libreria associato a un utente specifico.

aggiungiLibro()

Aggiunge un nuovo titolo di libro alla lista dei libri nella libreria.

toString()

Restituisce una rappresentazione testuale della libreria concatenando i valori di userID, nomeLibreria.

2.4 Libro:

Libro ()

Inizializza un nuovo oggetto Libro con i dettagli specificati riguardo al titolo, autori, anno di pubblicazione, editore e categoria.

ToString()

Restituisce una rappresentazione testuale dell'oggetto Libro concatenando i valori di titolo, autori, anno, editore e categoria.

2. 5 Trasferimento:

main()

Legge i dati dal file CSV specificato, li processa e li scrive nel file di output in un formato predefinito.

2.6 Utente:

Utente()

Inizializza un nuovo oggetto Utente con i dettagli personali e di autenticazione forniti. Assegna i valori forniti ai rispettivi campi (nomeCognome, codiceFiscale, email, userID, password) dell'oggetto Utente

2.7 Valutazione:

Valutazione()

Inizializza un nuovo oggetto Valutazione con i dettagli dell'utente, del libro e i punteggi per vari aspetti del libro. Assegna i valori forniti ai rispettivi campi (userID, titoloLibro, stile, contenuto, gradevolezza, originalità, edizione) dell'oggetto Valutazione.

calcolaVotoFinale()

Calcola il voto finale come la media aritmetica dei punteggi assegnati per i vari aspetti del libro.

toString()

Restituisce una rappresentazione testuale dell'oggetto Valutazione, che include l'ID utente, il titolo del libro, i punteggi assegnati e il voto finale.

3) Le scelte architetturali

In questa sezione verrano elencate e spiegate le scelte architetturali fatte per programmare l'applicazione: in particolare di seguito spieghiamo il significato delle principali classi scelte.

3.1 Lettura su file

• FileReader: La classe FileReader permette in maniera semplice di leggere il contenuto di un file (in questo caso formato .txt) e assosicato alla classe successiva semplifica anche la possibilità di leggerne il contenuto riga per riga.

• BufferedReader: per BufferedReader si intende la classe facente parte del pacchetto java.io che legge il contenuto di un file riga per riga, in questo caso ottimale ogni qual volta in cui è necessario controllare la correttezza delle credenziali di ciascun utente quando queste vengono salvate in una riga di codice unica per ogni nuovo utente.

3.2 Scrittura su file

• FileWriter: La classe in questione serve per scrivere su file di testo il contenuto di una variabile, o come in questo caso di un'insieme di variabili.

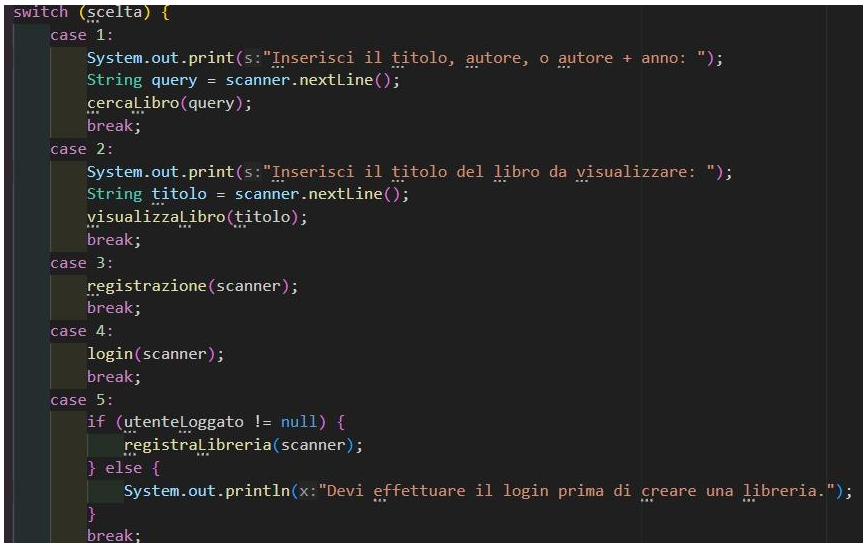
3.3 Input da tastiera

• Scanner: Il package java.util offre la classe Scanner che rappresenta un'ottima opzione per avere un supporto semplice da usare e da comprendere per l'input da tastiera da parte dell'utente. La scelta è ricaduta su questa classe perchè la classe ConsoleInputManager contenuta in un pack age custom ha causato numerosi problemi in fase di costituzione del file .jar per questioni di ClassPath.

4) Scelte algoritmiche

4.1 Switch:

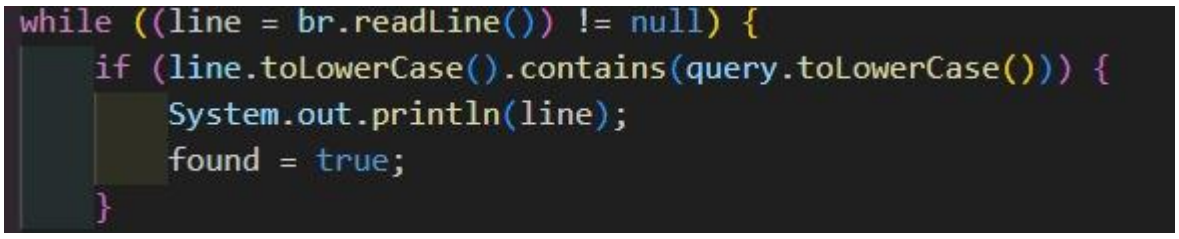
L'utilizzo dello switch è stato molto utile per la suddivisone di più richieste senza dover ogni volta regolare il tipo di domanda, questo permette di risparmiare linee di codice e rende il programma più efficiente e comprensibile:



```
switch (scelta) {
    case 1:
        System.out.print(s:"Inserisci il titolo, autore, o autore + anno: ");
        String query = scanner.nextLine();
        cercaLibro(query);
        break;
    case 2:
        System.out.print(s:"Inserisci il titolo del libro da visualizzare: ");
        String titolo = scanner.nextLine();
        visualizzaLibro(titolo);
        break;
    case 3:
        registrazione(scanner);
        break;
    case 4:
        login(scanner);
        break;
    case 5:
        if (utenteLoggato != null) {
            registraLibreria(scanner);
        } else {
            System.out.println(x:"Devi effettuare il login prima di creare una libreria.");
        }
        break;
```

4.2 While:

Il ciclo While serve per leggere i file ed il loro contenuto e verificare che esso non sia nullo, in particolare la variabile br riferita alla classe BufferedReader serve per controllare quanto appena detto. Nella foto un esempio:



```
while ((line = br.readLine()) != null) {
    if (line.toLowerCase().contains(query.toLowerCase())) {
        System.out.println(line);
        found = true;
    }
```

4.3 Try-catch:

Quando la funzione si basa sulla collaborazione di un file di testo (che si tratti di lettura o scrittura) viene eseguito un ciclo try-catch per effettuare tutte le istruzioni del caso e per tentare di afferrare (da qui catch) le eccezioni nel caso in cui si verifichino (in questo caso FileNotFoundException).

5) Strutture dati utilizzate

Per la realizzazione di questo progetto la scelta del supporto per le strutture dati è ricaduta sui file di testo per la loro innata facilià di essere modificati ed integrati con la classe BufferedReader e FileWriter.

I file di testo sono centrali nell'esecuzione e nella buona riuscita del programma.

E' molto importante che i file non vengano modificati a mano senza l'uso del programma in quanto potrebbero risultare corrotte o danneggiate alcune credenziali. Di seguito verranno elencati e spiegati i file di testo usati nel programma:

-Consigli.dati:

In questo file vengono riportati tutti i consigli scritti dagli utenti riportando il loro Username e il titolo del libro

-Librerie.dati:

Questo file contiene tutte le librerie personalizzate create dagli utenti nel proprio account

-Libri.dati:

La totalità delle informazioni dei libri che può utilizzare l'utente tra cui il nome dell'autore, le informazioni sul libro e il titolo

-UtentiRegistrati.dati:

Il file contente tutti i dati sensibili degli utenti registrati all'app.

-ValutazioneLibri.dati:

Il file contente tutte le valutazioni date dagli utenti con in aggiunta un voto finale che sta a decretare il livello di bellezza del libro.