



**UNIVERSITÀ DEGLI STUDI  
DELL'INSUBRIA**

# **Book Recommender**

## **Technical Manual**

**Laurea Triennale di Informatica**

**Progetto Laboratorio A: Book Recommender**

**Sviluppato da: Matteo Corda, Jacopo Loni, Simone Cirillo, Gabriele Schioppa e Simone Diano**

## Sommario:

Introduzione.....	2
Librerie utilizzate.....	2
<i>Struttura generale del sistema di classi</i> .....	2
Classi.....	3
<i>Utente</i> .....	3
<i>Autore</i> .....	4
<i>Libro</i> .....	4
<i>Librerie</i> .....	5
<i>ValutazioniLibro</i> .....	5
<i>SuggerimentiLibro</i> .....	6
<i>HomeMainFrame</i> .....	7
<i>LgMainFrame</i> .....	9
<i>RgMainFrame</i> .....	10
<i>LibrerieMainFrame</i> .....	11
Bibliografia.....	12

## Introduzione

Book Recommender è un progetto sviluppato per il progetto di Laboratorio interdisciplinare A per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria.

Il progetto è sviluppato in Java 12, usa un'interfaccia grafica costruita con Java Swing ed è stato sviluppato e testato sul sistema operativo Windows 10.

### *Librerie utilizzate:*

*Per lo sviluppo di questo progetto sono state utilizzate alcune librerie tra cui:*

- **Java.swing:** contiene tutti gli elementi per lo sviluppo dell'interfaccia grafica.
- **Java.File:** utilizzata per la creazione e la gestione dei file.
- **Java.io:** utilizzate per le operazioni di scrittura e lettura dei file.
- **Java.util:** sono state utilizzate alcune librerie di Java.util come ArrayList e Scanner.
- **Java.awt:** utilizzata per la gestione degli eventi, inoltre contiene GridBagConstraints che permette di gestire la struttura della GUI con delle griglie.

### *Struttura generale del sistema di classi*

*Il progetto presenta una suddivisione delle classi:*

- **classi utilizzate per la creazione e gestione di oggetti** che si occupano della vera e propria elaborazione dei dati:

<b>Utente</b>	Per la creazione e la gestione degli utenti
<b>Autore</b>	Per la creazione e la gestione degli autori
<b>Libro</b>	Per la creazione e la gestione dei libri
<b>Librerie</b>	Per la creazione e la gestione delle librerie
<b>ValutazioniLibro</b>	Per aggiungere le valutazioni al libro
<b>SuggerimentoLibro</b>	Per aggiungere dei suggerimenti

- **classi adibite alla gestione dell'interfaccia grafica.**

<b>HomeMainFrame</b>	Gestisce la schermata principale
<b>LgMainFrame</b>	Gestisce la schermata di Login
<b>RgMainFrame</b>	Gestisce la schermata di Registrazione
<b>LibrerieMainFrame</b>	Gestisce le librerie con tutte le loro funzioni

*Ora vedremo nel dettaglio le singole classi, con strutture dati e algoritmi utilizzati:*

## Utente

---

La classe utente implementa l'**interfaccia Serializable**, utilizzate per la serializzazione di un ArrayList di utenti all'interno del file **Utenti.txt**.

*La classe presenta i seguenti parametri:*

- **nome**: String che rappresenta il nome dell'utente
- **cognome**: String che rappresenta il cognome dell'utente
- **Email**: String che rappresenta la mail dell'utente
- **Username**: String che rappresenta l'username scelto dall'utente
- **Password**: String che rappresenta la password scelta dall'utente
- **Codice**: Intero utilizzato per il controllo di eventuali errori
- **AlUtente**: ArrayList contenente gli utenti

*Sono contenuti diversi metodi:*

- **stringheVuote**: prende in input 5 stringhe ovvero nome, cognome, email, username, password e restituisce TRUE se almeno una delle stringhe è vuota, FALSE altrimenti.
- **controlloEmail**: prende in input una stringa (Email) e restituisce FALSE se presenta "@", TRUE altrimenti
- **leggiFile**: metodo void che riempie l'ArrayList AlUtente in seguito alla lettura del file Utenti.txt
- **emailEsistente**: prende in input l'ArrayList AlUtente e la stringa Email e restituisce TRUE se esiste già la stessa Email tra gli utenti registrati, FALSE altrimenti.
- **usernameEsistente**: prende in input l'ArrayList AlUtente e la stringa Username e restituisce TRUE se esiste già lo stesso username tra gli utenti registrati, FALSE altrimenti.
- **Metodi Get e Set** di ogni parametro descritto sopra.
- **Costruttore parametrico**: prende in input tutti i parametri (tranne codice); all'interno avviene il controllo degli errori. Il codice viene settato a 4 se il metodo stringheVuote restituisce TRUE ed inseguito fa un RETURN. Viene settato a 5 se invece il controllo EMAIL è uguale a TRUE, ed esegue un RETURN. Viene settato a 0 se la Email è già esistente. Viene settato a 1 se l'username è già esistente. Restituisce 0 se la creazione va a buon fine.

## Autore

---

Anch'essa implementa **Serializable**, in questo caso utilizzata per la serializzazione dei libri.

*La classe presenta il seguente parametro:*

- **nome**: stringa che rappresenta il nome dell'autore

*La classe presenta due metodi:*

- **costruttore parametrico**: prende in input il nome ed inizializza l'oggetto di tipo autore.
- **getNome**: restituisce il nome dell'autore

## Libro

---

La classe libro implementa sia **Serializable** che **Cloneable**, la prima viene utilizzata per serializzare un ArrayList di libri all'interno di un file; la seconda invece per "clonare" un oggetto di tipo libro.

*La classe presenta i seguenti parametri:*

- **titolo**: è una stringa che rappresenta il titolo del libro
- **autore**: campo di tipo Autore che identifica l'autore
- **descrizione**: Stringa che identifica la descrizione del libro
- **categoria**: Stringa che identifica la categoria del libro
- **editore**: Stringa che identifica l'editore del libro
- **dataPubblicazione**: LocalDate che rappresenta la data di pubblicazione del libro
- **prezzo**: double che identifica il prezzo del libro

*La classe presenta i seguenti metodi:*

- **costruttore parametrico**: prende in input tutti i parametri sopracitati ed inizializza un oggetto di tipo libro
- **metodi get e set**: di ogni parametro
- **Metodo toString**: restituisce una stringa contenente tutte le informazioni del libro
- **visualizzaLibro**: metodo statico che prende in input un libro e ne restituisce le informazioni, richiama il toString().
- **Clone**: metodo che "clona" il libro che richiama il metodo
- **getInfoBase**: metodo statico che prende in input un libro e ne restituisce le informazioni formattate tramite i tag HTML.

## Librerie

---

La classe librerie implementa l'interfaccia **Serializable**, utilizzate per la serializzazione di un ArrayList di utenti all'interno del file "Librerie.txt".

*La classe presenta i seguenti parametri:*

- **nome:** String che rappresenta il nome della libreria
- **utente:** Oggetto utente che rappresenta l'utente riferito alla libreria
- **alLibri:** ArrayList di tipo Libro che contiene i libri della libreria

*Sono contenuti diversi metodi:*

- **aggiungiLibro:** prende in input un oggetto di tipo Libro e lo aggiunge all'ArrayList alLibro, non restituisce niente
- **scriviAlLibrerieFile:** scrive l'ArrayList contenente le librerie dentro ad un file "Librerie.txt" per salvarli, metodo void static
- **filtraLibrerie:** restituisce un ArrayList che contiene tutte le librerie di un utente specifico, l'utente viene dato come parametro al metodo
- **Metodi Get** di ogni parametro
- **Costruttore Parametrico:** prende in input tutti i parametri e li salva nella classe.

## ValutazioniLibro

---

La classe ValutazioniLibro implementa Serializable per serializzare l'ArrayList di valutazioniLibro all'interno di un file .txt

*La classe presenta i seguenti parametri:*

- **utente:** oggetto di tipo utente a cui è associata la valutazione
- **libro:** oggetto di tipo libro a cui è associata la valutazione

qui di seguito sono riportati i parametri per la valutazione del libro, si dividono in voti e note, ovvero i commenti riguardanti quel campo:

- stile e noteStile
- contenuto e noteContenuto
- gradevolezza e noteGradevolezza
- originalità e noteOriginalità
- edizione e noteEdizione

- votoFinale noteVotoFinale

*La classe presenta i seguenti metodi:*

- **costruttore parametrico:** prende in input utente e il libro ed inizializza un oggetto di tipo valutazioneLibro
- **inserisciValutazioneLibro:** metodo void che prende in input i campi voti e note settandone il valore
- **get e set:** metodi get e set di qualsiasi parametro
- **scriviFileVL:** metodo statico che prende in input un ArrayList di valutazioniLibro e lo scrive sul file valutazioni.txt
- **leggiFileVL:** metodo statico che legge il file valutazioni.txt e restituisce un ArrayList di tipo valutazioniLibro

## SuggerimentoLibro

---

La classe suggerimentoLibro implementa Serializable per serializzare l'ArrayList di suggerimentoLibro all'interno di un file .txt

*La classe presenta i seguenti parametri:*

- **libro:** oggetto di tipo libro a cui sono associati i suggerimenti
- **utente:** oggetto di tipo utente a cui sono associati i suggerimenti
- **suggerimenti:** ArrayList che contiene i suggerimenti

*La classe presenta i seguenti metodi:*

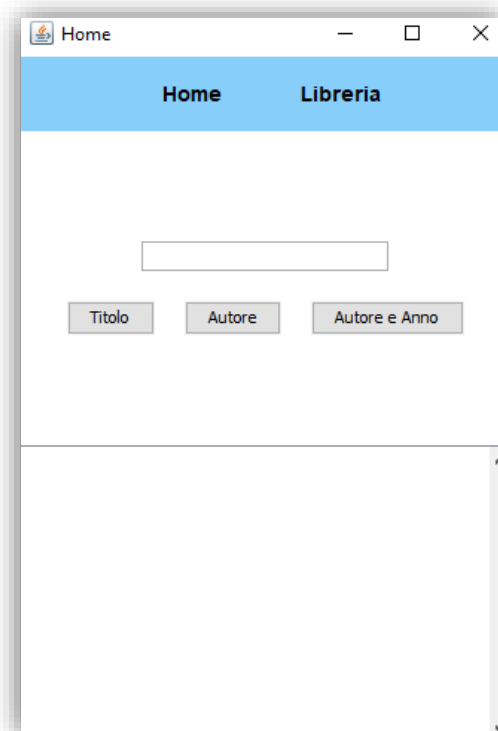
- **costruttore parametrico:** prende in input il libro e l'utente, inizializza l'ArrayList suggerimenti ed inizializza un oggetto di tipo SuggerimentoLibro
- **metodi get:** metodi get dei 3 parametri
- **scriviFileSugg:** metodo statico che prende in input un ArrayList di suggerimentiLibro e lo scrive sul file suggerimenti.txt
- **leggiFileSugg:** metodo statico che legge il file suggerimenti.txt e restituisce un ArrayList di tipo suggerimentoLibro.
- **inserisciSuggerimento:** metodo void che prende in input un ArrayList di suggerimenti e lo salva all'interno dell'ArrayList dell'oggetto

*di seguito invece sono riportate le classi per la gestione e la creazione della GUI:*

## HomeMainFrame

---

Gestisce la pagina principale denominata Home, è strutturata con un top panel contenente due pulsanti: Home e Libreria, al centro dell'interfaccia è presente un textfield con associati 3 pulsanti per la ricerca, infine nella parte finale è presente un pannello dove verranno mostrati i risultati.



La classe presenta diversi parametri per la creazione e il corretto funzionamento dell'interfaccia.

*La classe presenta i seguenti metodi:*

- **Costruttore parametrico:** prende in input un'ArrayList di libri ovvero tutti i libri contenuti all'interno del file CSV che abbiamo estratto e un utente che rappresenta l'attuale utente che ha effettuato il login e un codice che può variare tra 0 e 1 in base al tipo di login effettuato (Username e Password oppure Guest Access). All'interno del costruttore inoltre vengono inizializzati la maggior parte dei parametri come ad esempio i Frame che compongono la GUI ed i vari bottoni. Infine è presente la logica dietro al bottone "libreria":

inizializza il frame LibrerieMainFrame che vederemo nel dettaglio più avanti, chiudendo il frame corrente se il codice è uguale ad 1, se invece è uguale a 0 mostra un messaggio che dichiara l'impossibilità di effettuare l'operazione



vengono anche inizializzati i pulsanti ed i label di ricerca con il loro funzionamento associato.

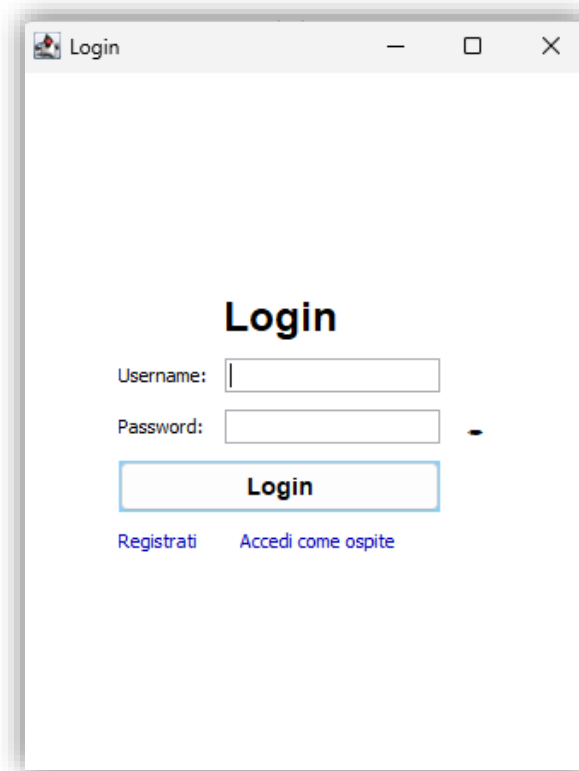
Vengono gestiti i 3 pulsanti per la ricerca:

- il pulsante autore ed il pulsante titolo salvano in una stringa il contenuto del label di ricerca e se il contenuto non è vuoto viene effettuata la ricerca tramite il metodo `cercaLibro()` e ne mostra i risultati all'interno del label sottostante tramite il metodo `mostraRisultati()`.
  - Il pulsante autore e anno invece effettua la stessa operazione preliminare ma prima di effettuare la ricerca, crea un Array formato dalle due parti (autore – anno) e tramite esso effettua la ricerca
- 
- **CustomizeButton**: metodo void privato che prende in input un bottone e ne modifica l'estetica (BackgroundColor, padding...)
  - **Initialize**: metodo pubblico che permette la visione del frame settandone il titolo e le dimensioni
  - **Metodi cercaLibro**: metodi privati che restituiscono un ArrayList di tipo libro, tutti e 3 effettuano la stessa operazione ma prendendo in input parametri diversi
  - **mostraRisultati**: metodo privato che mostra all'interno del label del risultato tutti i libri presenti all'interno dell'ArrayList ottenuto tramite il metodo `cercaLibro`; il metodo quando viene richiamato ripulisce il pannello dei risultati ed in seguito grazie all'utilizzo di un ciclo FOR crea tanti label quanti sono i libri contenuti nell'ArrayList. Ogni label mostrato è cliccabile e mostra le informazioni del libro (tramite `mostraDettagliLibro()` )
  - **mostraUtentiRecensioni**: metodo privato che prende in input un ArrayList di utenti e un libro e tramite un FOR che scorre tutto l'ArrayList, vengono creati tanti label quanti gli utenti che hanno lasciato una recensione al libro, ogni label è a sua volta cliccabile e mostra le valutazioni dell'utente
  - **mostraDettagliLibro**: metodo privato che prende in input un libro e un intero n; questo metodo mostra le informazioni di tutto il libro comprese la media delle valutazioni di tutti gli utenti, n viene utilizzato per includere o escludere alcune informazioni come la lista degli utenti che hanno lasciato delle valutazioni
  - **mediaValutazioniLibri**: metodo privato che prende in input un libro e restituisce la media di tutti i tipi di valutazione, un ArrayList contenente gli utenti che hanno lasciato una valutazione al libro e un ArrayList di utenti che hanno lasciato almeno un suggerimento al libro.
  - **mostraValUtente**: metodo privato che prende in input un utente ed un libro; questo metodo viene utilizzato all'interno di altri metodi per mostrare le valutazioni ed i suggerimenti dell'utente al singolo libro

## LgMainFrame

---

La classe LgMainFrame serve per la rappresentazione dell'interfaccia grafica della finestra di login del programma, presenta un textfield per l'username e uno per la password, seguiti da un tasto "login", sono presenti anche due tasti "registrati" ed "accedi come ospite", che sono altre opzioni di accesso disponibili.



La classe presenta diversi parametri per la creazione e il corretto funzionamento dell'interfaccia grafica.

*Sono contenute diversi metodi:*

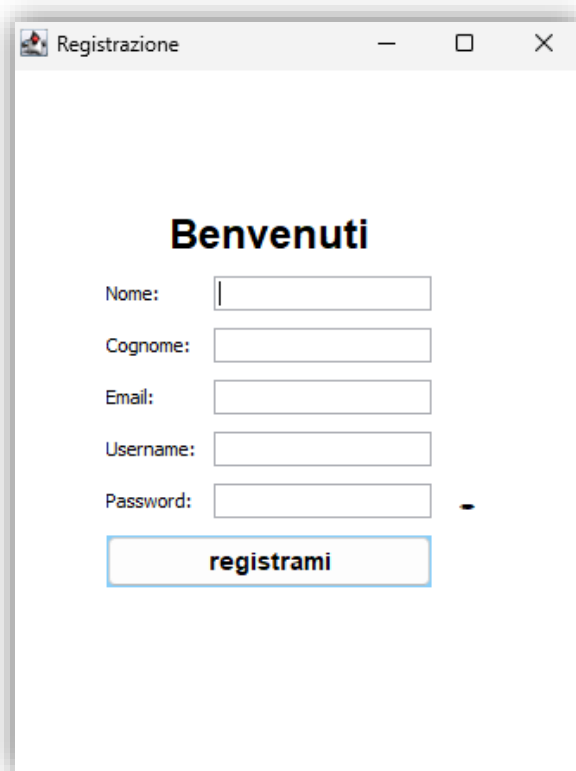
- **initialize:** inizializza il frame impostando alcuni aspetti come dimensione del frame e modalità di chiusura
- **autenticazione:** prende in input due String, l'username e la password e controlla che siano corrette ovvero se sono presenti nel file che contiene le credenziali. Ritorna la posizione dell'utente nell'ArrayList se le credenziali sono corrette, altrimenti ritorna -1
- **leggiFile:** legge dal file "utenti.txt" gli utenti e restituisce un ArrayList di tipo Utente contenente tutti gli utenti registrati
- **Metodo Get** del parametro String password
- **Costruttore Parametrico:** prende in input l'ArrayList contenente tutti i libri precedentemente estratti dal "BooksDataset.csv". Il costruttore permette di costruire il frame di login dell'applicazione, contiene quindi tutti i parametri e metodi che permettono di strutturare e personalizzare a proprio modo l'interfaccia grafica di questo frame

## RgMainFrame

---

La classe RgMainFrame serve per la rappresentazione dell'interfaccia grafica della finestra di registrazione del programma. Sono presenti i vari textfield con i dati da inserire per la registrazione ed un tasto per confermare la registrazione

La classe presenta diversi parametri per la creazione e il corretto funzionamento dell'interfaccia grafica.



*Sono contenuti diversi metodi:*

- **initialize:** inizializza il frame impostando alcuni aspetti come dimensione del frame e modalità di chiusura
- **leggiFile:** legge dal file "utenti.txt" gli utenti e restituisce un ArrayList di tipo Utente contenente tutti gli utenti registrati
- **Metodo Get** del parametro String password
- **Costruttore Parametrico:** prende in input l'ArrayList contenente tutti i libri precedentemente estratti dal "BooksDataset.csv". Il costruttore permette di costruire il frame di registrazione dell'applicazione, contiene quindi tutti i parametri e metodi che permettono di strutturare e personalizzare a proprio modo l'interfaccia grafica di questo frame.

## LibrerieMainFrame

---

La classe LibrerieMainFrame serve per la rappresentazione dell'interfaccia grafica della finestra di gestione delle librerie del programma.

La classe presenta diversi parametri per la creazione e il corretto funzionamento dell'interfaccia grafica.

*Sono contenuti diversi metodi:*

- **initialize:** inizializza il frame impostando alcuni aspetti come dimensione del frame e modalità di chiusura
- **fileEsiste:** Verifica l'esistenza di un file dato in input come parametro di tipo file. Se non esiste lo crea.
- **Costruttore Parametrico:** prende in input l'ArrayList contenente tutti i libri precedentemente estratti dal "BooksDataset.csv" e l'utente. Il costruttore permette di costruire il frame di login dell'applicazione, contiene quindi tutti i parametri e metodi che permettono di strutturare e personalizzare a proprio modo l'interfaccia grafica di questo frame
- **listaLibrerie:** Metodo void che prende in input un ArrayList contenente l'elenco delle librerie e le mostra all'utente nel frame.
- **customizeButton:** metodo void privato che prende in input un bottone e ne modifica l'estetica (BackgroundColor, padding...)
- **mostraInserimentoValutazioni:** Metodo void che mostra un dialogo per inserire o modificare le valutazioni di un libro preso in input da parte di un utente.
- **mostraRisultati:** metodo void che prende in input un numero che passa al metodo mostraInfoInLibrerie, un ArrayList di libri risultati della ricerca all'interno di un pannello specifico, anch'esso dato in input.
- **mostraInfoInLibrerie:** Metodo void che prende in input un numero e un libro. Se il numero è uguale a zero consente all'utente di aggiungere il libro alla propria libreria altrimenti permette di aggiungere il libro ai suggerimenti.
- **cercaLibro:** tre metodi privati che restituiscono un ArrayList di tipo libro, tutti e 3 effettuano la stessa operazione ma prendendo in input parametri diversi
- **Suggerimenti:** Metodo void che mostra l'interfaccia utente per inserire suggerimenti per un libro specifico.
- **Opzioni:** Metodo void che apre una finestra da cui l'utente può selezionare se vuole suggerire dei libri correlati al libro preso in input o se vuole valutare il libro
- **ricercaConAutore:** Esegue una ricerca di libri nell'ArrayList contenente tutti i libri basata sull'autore dato in input come String. Restituisce l'ArrayList contenente tutti i libri risultati dalla ricerca.

- **ricercaConAnnoAutore:** Esegue una ricerca di libri nell'ArrayList contenente tutti i libri basata sull'anno e l'autore dati in input entrambi come un'unica String. Restituisce l'ArrayList contenente tutti i libri risultati dalla ricerca.
- **ricercaConTitolo:** Esegue una ricerca di libri nell'ArrayList contenente tutti i libri basata sul titolo dato in input come String. Restituisce l'ArrayList contenente tutti i libri risultati dalla ricerca.
- **AggiungiBottoniECampoRicerca:** Metodo void che aggiunge i componenti grafici per la ricerca (bottone Titolo, Autore, Autore e Anno e campo di testo) al pannello specificato in input.

## Bibliografia:

---

**Java.swing** - <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

**Java.file** - <https://docs.oracle.com/javase/8/docs/api/java/io/File.html>

**Java.io** - <https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html>

**Java.util** - <https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html>

**Java.awt** - <https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html>