DESCRIZIONE DELLE CLASSI-GESTIONE BIBLIOTECA

Descrizione delle Classi - Sistema Gestione Biblioteca

Classe 'Libro'

Questa classe rappresenta un singolo libro nella collezione della biblioteca, includendo tutti gli attributi ad esso associati, quali:

- ISBN (codice identificativo univoco);
- titolo;
- autori (lista);
- anno di pubblicazione;
- copie disponibili;
- copie totali.

La classe implementa anche dei metodi:

- getter e setter per tutti gli attributi del libro. In questo modo è più semplice creare un nuovo libro, assegnargli valori specifici a ciascun campo o recuperare i dati di un libro già esistente. Questi metodi, insieme all'utilizzo degli attributi privati, permettono l'implementazione dell'incapsulamento;
- AggiungiAutore() per inserire il nome e il cognome dell'autore del libro;
- isDisponibile(), per verificare se almeno una copia del libro è disponibile;
- decrementaCopie(), per ridurre il numero di copie disponibili quando il libro viene prestato;
- incrementaCopie(), per aumentare il numero di copie disponibili quando il libro viene restituito;
- toString(), per visualizzare le informazioni relative al libro.

Classe 'Utente'

Questa classe rappresenta un singolo utente della biblioteca, includendo tutti gli attributi ad esso associati, quali:

- nome;
- cognome;
- email;
- matricola;

- prestiti attivi (lista);

  La classe implementa anche dei metodi:

- getter e setter per tutti gli attributi dell'utente. Questi metodi permettono l'incapsulamento;
- haRaggiuntoLimite() verifica se l'utente ha raggiunto il numero massimo di prestiti attivi.
- aggiungiPrestito(), per aggiungere un prestito alla lista dei prestiti attivi dell'utente;
- rimuoviPrestito(), per rimuovere un prestito dalla lista dei prestiti attivi quando viene restituito;
- getPrestitiAttivi(), per ottenere la lista dei prestiti attualmente attivi;
- getNumeroPrestiti(), per contare il numero totale di prestiti attivi;
- toString(), per visualizzare le informazioni relative all'utente.

Classe 'Prestito'

Questa classe rappresenta un singolo prestito effettuato da un utente per un libro, includendo tutti gli attributi ad esso associati, quali:

- utente (riferimento all'utente);
- libro (riferimento al libro prestato);
- data prestito;
- data prevista di restituzione;
- data effettiva di restituzione;
- giorni di ritardo.
- id

La classe implementa anche dei metodi:

- getter e setter per tutti gli attributi del prestito. Questi metodi permettono l'incapsulamento;
- generateId() genera un indirizzo id univoco;
- isAttivo() per verificare se un prestito è in attivo;
- registrareRestituzione(), per registrare la data effettiva di restituzione del prestito;
- isInRitardo(), per verificare se il prestito è in ritardo rispetto alla data prevista;

- getGiorniRitardo(), per ottenere il numero di giorni di ritardo;
- registraRitardo(), per registrare i giorni di ritardo accumulati;
- toString(), per visualizzare le informazioni relative al prestito.

Classe 'Biblioteca'

La classe Biblioteca gestisce la collezione completa di libri, utenti e prestiti della biblioteca. La classe ha i seguenti attributi:

- List<Libro> libri: una lista osservabile che contiene tutti i libri presenti nella biblioteca;
- List<Utente> utenti: una lista osservabile che contiene tutti gli utenti registrati;
- List<Prestito> prestiti: una lista osservabile che contiene tutti i prestiti effettuati;
- List<Prestito> prestitiAttivi: una lista osservabile che contiene i prestiti attualmente attivi.
- Static Biblioteca instance: evita duplicazioni di dati

La classe implementa i seguenti metodi:

- aggiungiLibro(), consente di aggiungere un nuovo libro alla collezione della biblioteca;
- modificaLibro(), consente di modificare gli attributi di un libro già presente;
- eliminaLibro(), consente di rimuovere un libro dalla collezione;
- cercaLibroPerISBN(), effettua una ricerca di un libro in base all'ISBN;
- cercaLibri(), effettua una ricerca di libri, che può essere in base al titolo, all'autore e/o all'ISBN.
- aggiungiUtente(), consente di aggiungere un nuovo utente alla collezione;
- modificaUtente(), consente di modificare gli attributi di un utente già presente;
- eliminaUtente(), consente di rimuovere un utente dalla collezione;
- cercaUtenti(), effettua una ricerca di un utente in base ai dati dell'utente;
- cercaUtentePerMatricola() effettua una ricerca di un utente in base alla sua matricola;
- registraPrestito(), consente di registrare un nuovo prestito;
- registraRestituzione(), consente di registrare la restituzione di un prestito;
- getPrestitiAttivi(), restituisce la lista dei prestiti attualmente attivi;
- salvaDati(), salva tutti i dati (libri, utenti, prestiti) su file oppure database;

- caricaDati(), carica i dati da file oppure database;
- getTuttiLibri(), getUtenti(), getPrestiti(), restituiscono le rispettive liste.

Classe 'MainApp'

La classe MainApp è il punto di ingresso principale dell'applicazione. Questa classe è responsabile dell'inizializzazione dell'applicazione e dell'avvio della finestra principale.

La classe implementa i seguenti metodi:

- main(), metodo statico che avvia l'applicazione;
- start(), inizializza e avvia la finestra principale della GUI.
- Stop() serve a terminare l'app, ioè chiudere l'applicazione

---

Classe 'MainController'

La classe MainController è il controller principale che orchestra l'intera applicazione e gestisce la navigazione tra i diversi moduli (gestione libri, gestione utenti, gestione prestiti). La classe contiene mainBorderPane che, annotato in FXML, rappresenta il contenitore principale dell'interfaccia grafica.

La classe implementa i seguenti metodi:

- void initialize(), inizializza il controller principale e carica i dati dalla biblioteca;
- void apriGestioneLibri(), apre il modulo di gestione dei libri;
- void apriGestioneUtenti(), apre il modulo di gestione degli utenti;
- void apriGestionePrestiti(), apre il modulo di gestione dei prestiti;
- void mostraInformazioni(), mostra le informazioni generali della biblioteca;
- void caricaSchermata(), carica e visualizza la schermata principale;
- void esci(), chiude l'applicazione in modo sicuro salvando i dati.

Classe 'LibroController'

La classe LibroController è il controller responsabile della gestione di tutte le operazioni relative ai libri (inserimento, modifica, eliminazione, visualizzazione). La classe ha i seguenti attributi:

- Biblioteca biblioteca: rappresenta la collezione di libri;

- ObservableList<Libro> lista Libri, contiene e mostra la lista di libri

La classe contiene, inoltre, dei campi annotati con FXML, che rappresentano e gestiscono vari elementi dell'interfaccia utente:

- TableView, tabella in cui vengono visualizzati i libri;
- TableColumn, ognuna delle quali rappresenta una colonna nella tabella con i dati inerenti ai libri
- TextField, barra di ricerca per i libri;

La classe implementa i seguenti metodi:

- void initialize(), inizializza il controller e carica i libri dalla biblioteca;
- void apriDialogoNuovoLibro(), apre il dialogo per inserire un nuovo libro;
- void aggiornaTabella(), aggiorna la TableView con i dati attuali della collezione;
- void mostraTuttiLibri(), mostra tutti i libri all'utente;
- void modificaLibro() permette di modificare i dati di un libro;
- void inserisciLibro() inserisce un libro nella tabella;
- void eliminaLibro() elimina un libro dalla tabella;
- void cercaLibrio() cerca un libro all'interno della tabella;

Classe 'UtenteController'

La classe UtenteController è il controller responsabile della gestione di tutte le operazioni relative agli utenti (inserimento, modifica, eliminazione, visualizzazione). La classe ha i seguenti attributi:

- Biblioteca biblioteca: rappresenta la collezione di utenti;
- ObservableList <Utente> mostra la lista degli utenti;

  La classe contiene, inoltre, dei campi annotati con FXML, che rappresentano e gestiscono vari elementi dell'interfaccia utente:

- TableView, tabella in cui vengono visualizzati gli utenti;
- TextField, barra di ricerca per gli utenti;
- TableColumn, ognuna delle quali rappresenta una colonna nella tabella con i dati inerenti agli utenti

La classe implementa i seguenti metodi:

- void initialize(), inizializza il controller e carica gli utenti dalla biblioteca;
- void apriDialogoNuovoUtente(), apre il dialogo per inserire un nuovo utente;
- void aggiornaTabella(), aggiorna la TableView con i dati attuali della collezione;
- void mostraTuttiUtenti(), mostra tutti gli utenti (studenti) all'utente;
- void modificaUtente() permette di modificare i dati di un utente;
- void inserisciUtente() inserisce un utente nella tabella;
- void eliminaUtente() elimina un utente dalla tabella;
- void cercaUtente() cerca un utente all'interno della tabella;

Classe 'PrestitoController'

La classe PrestitoController è il controller responsabile della gestione di tutte le operazioni relative ai prestiti (registrazione, restituzione, visualizzazione dello stato). La classe ha il seguente attributo:

- Biblioteca biblioteca: rappresenta la collezione di prestiti.
- ObservableList <Prestito> mostra la lista dei prestiti attivi.

La classe contiene, inoltre, dei campi annotati con FXML, che rappresentano e gestiscono vari elementi dell'interfaccia utente:

- TableView, tabella in cui vengono visualizzati i prestiti attivi;
- TableColumn, ognuna delle quali rappresenta una colonna nella tabella con i dati inerenti ai prestiti
- LabelTotale, etichetta che mostra il numero totale di prestiti attivi in biblioteca
- LabelRitardi, etichetta che mostra il numero di prestiti in ritardo
- LabelScadenza, etichetta che mostra i prestiti in scadenza prossima

La classe implementa i seguenti metodi:

- void registraPrestito(), registra un nuovo prestito tra un utente e un libro;
- void initialize() inizializza il controller e carica i prestiti nella tabella
- void registraRestituzione(), registra la restituzione di un prestito e traccia i ritardi;
- void aggiornaTabella(), aggiorna la TableView con i prestiti attivi;

- void aggiornaStatistiche() aggiorna le etichette informative con i nuovi dati

---

Classe 'LibroDialogController'

La classe LibroDialogController gestisce la logica per la finestra di dialogo utilizzata per aggiungere o modificare un libro. La classe ha i seguenti attributi:

- TextField fieldTitolo, Campo testo per il titolo del libro
- TextField fieldIsbn, Campo testo per l'ISBN
- TextField fieldAnno, Campo testo per l'anno di pubblicazione
- TextField fieldCopie, Campo testo per il numero di copie
- ListView<Libro.Autore> listaAutori, Lista visualizzabile degli autori
- TextField fieldNomeAutore, Campo testo per il nome dell'autore
- TextField fieldCognomeAutore, Campo testo per il cognome dell'autore
- Biblioteca biblioteca, Riferimento alla biblioteca
- Libro libroCorrente, Il libro attualmente in modifica
- boolean confermato, Flag che indica se i dati sono stati confermati
- ObservableList<Libro.Autore> autori, Lista osservabile degli autori

La classe implementa i seguenti metodi:

- void inizializza(), Inizializza il dialog
- void setLibro(Libro libro), Imposta il libro da modificare
- void aggiungiAutore(), Aggiunge un autore alla lista
- void rimuoviAutore(), Rimuove un autore dalla lista
- void conferma(), Conferma i dati inseriti
- void annulla(), Annulla le modifiche
- void chiudi(), Chiude il dialog
- boolean isConfermato(), Restituisce se è stato confermato

Classe 'UtenteDialogController'

La classe UtenteDialogController gestisce la logica per la finestra di dialogo utilizzata per aggiungere o modificare un utente. La classe ha i seguenti attributi:

- TextField fieldNome, Campo testo per il nome dell'utente
- TextField fieldCognome, Campo testo per il cognome dell'utente
- TextField fieldMatricola, Campo testo per il numero di matricola (numero tessera)
- TextField fieldEmail, Campo testo per l'email dell'utente
- Label labelPrestitiAttivi, Etichetta che mostra il numero di prestiti attivi dell'utente
- Label labelStato, Etichetta che mostra lo stato dell'utente
- Biblioteca biblioteca, Riferimento alla biblioteca
- Utente utenteCorrente, L'utente attualmente in modifica
- boolean confermato, Flag che indica se i dati sono stati confermati

La classe implementa i seguenti metodi:

- void inizializza(), Inizializza il dialog
- void setUtente(Utente utente), Imposta l'utente da modificare
- void conferma(), Conferma i dati inseriti
- void annulla(), Annulla le modifiche
- void chiudi(), Chiude il dialog
- boolean isConfermato(), Restituisce se è stato confermato

Classe 'AlertHelper'

La classe AlertHelper è una classe utility che fornisce metodi statici per la visualizzazione standardizzata di messaggi di errore, avviso e informazione all'utente. Questa classe non ha attributi di istanza, poiché tutti i metodi sono statici.

La classe implementa i seguenti metodi:

- static void mostraErrore(String titolo, String messaggio), Mostra una finestra di dialogo di errore con titolo e messaggio personalizzati
- static void mostraConferma(String titolo, String messaggio), Mostra una finestra di dialogo di conferma con titolo e messaggio personalizzati
- static void mostraInfo(String titolo, String messaggio), Mostra una finestra di dialogo informativa con titolo e messaggio personalizzati

- static boolean mostraConfermaCancellazione(String oggetto), Mostra una finestra di dialogo di conferma per la cancellazione di un oggetto e restituisce true se l'utente conferma, false altrimenti