Analisi di coesione e accoppiamento

Di seguito sono presenti le tabelle che analizzano coesione e accoppiamento di varie classi.

Classe	COESIONE
MainApp	Funzionale
GIUSTIFICAZIONE	
La classe è responsabile di un singolo compito, ovvero gestire il ciclo di vita dell'applicazione JavaFX. Tutti i metodi e i campi sono strettamente correlati a questo obiettivo: - start() inizializza l'interfaccia grafica e carica i dati. - stop() salva i dati alla chiusura. - L'attributo biblioteca mantiene il riferimento al model per le operazioni di caricamento/salvataggio.	

Classe	COESIONE
Libro	Funzionale
GIUSTIFICAZIONE	
La classe è responsabile di un singolo compito, ovvero rappresentare un libro del catalogo e gestire la disponibilità delle copie. Ogni metodo è strettamente correlato a questo obiettivo: - incrementaCopie() e decrementaCopie() gestiscono la disponibilità delle copie durante prestiti/restituzioni. - isDisponibile() verifica se esistono copie disponibili. - getAutoriAsString() formatta i dati per la visualizzazione della lista autori. - Gli attributi (isbn, titolo, autori, anno, copie) descrivono completamente l'entità libro.	

Classe	COESIONE
Libro.Autore	Funzionale
GIUSTIFICAZIONE	
La classe annidata rappresenta un'unica entità (autore) con nome e cognome. Tutti i metodi operano su questa entità: - I metodi getter restituiscono i dati dell'autore. - toString() formatta l'autore per la visualizzazione ("Cognome Nome"). - Gli attributi (nome, cognome) definiscono completamente l'autore. La classe è annidata perché gli autori esistono sempre nel contesto di un libro.	

Classe	COESIONE
Utente	Funzionale
GIUSTIFICAZIONE	
La classe è responsabile di gestire i dati e le operazioni relative a un singolo utente della biblioteca. Ogni metodo è correlato a questo obiettivo: - haRaggiuntoLimite(), aggiungiPrestito() e rimuoviPrestito() gestiscono la lista dei prestiti attivi dell'utente. - getNumeroPrestitiAttivi() e getNomeCognome() forniscono informazioni derivate sull'utente. - Gli attributi (matricola, nome, cognome, email, prestitiAttivi) descrivono completamente l'entità utente.	

Classe	COESIONE
Prestito	Funzionale

GIUSTIFICAZIONE
La classe gestisce un singolo compito ben definito: rappresentare e gestire il ciclo di vita di un prestito. Tutti i metodi operano su questo concetto: - isAttivo() e isInRitardo() getGiorniRitardo() e getGiorniAllaScadenza() verificano lo stato del prestito e implementano la logica temporale relativa. - registraRestituzione() chiude il prestito. - getStatoDescrizione() formatta lo stato per la UI. - Tutti i calcoli usano gli attributi del prestito (date, utente, libro).

Classe	COESIONE
Biblioteca	Funzionale
GIUSTIFICAZIONE	
La classe funge da punto di coordinamento per il sistema biblioteca, gestendo libri, utenti e prestiti in modo coordinato. Nonostante gestisca tre entità, mantiene coesione funzionale perché: - Tutti i metodi operano sul dominio "biblioteca universitaria". - Le operazioni su libri, utenti e prestiti sono correlate (ad esempio registraPrestito coinvolge tutti e tre). - La gestione della persistenza è centralizzata per garantire la consistenza dei dati.	

Classe	COESIONE
MainController	Procedurale
GIUSTIFICAZIONE	
La classe è responsabile di un unico compito: gestire la navigazione dell'interfaccia principale dell'applicazione in base alla selezione del menu. Tutti i metodi contribuiscono a questo obiettivo: - L'utente seleziona una voce di menu. - Il controller chiama il metodo corrispondente (apriGestioneLibri, apriGestioneUtenti, apriGestionePrestiti). - Il metodo privato caricaSchermata() carica il relativo FXML nel BorderPane centrale. - Il metodo initialize() imposta la schermata iniziale secondo la logica di navigazione.	

Classe	COESIONE
LibroController	Comunicazionale
GIUSTIFICAZIONE	
La classe è responsabile della gestione della tabella dei libri e coordina tutte le operazioni relative alla sua visualizzazione e modifica. I metodi lavorano sulla stessa struttura dati (la lista dei libri) ma svolgono compiti diversi: - inserisciLibro(), modificaLibro(), eliminaLibro() modificano la collezione dei libri. - cercaLibro() e mostraTuttiLibri() filtrano/visualizzano i dati. - aggiornaTabella() ricarica e ordina la TableView. - apriDialogLibro() gestisce l'interfaccia per le operazioni di inserimento e modifica.	

Classe	COESIONE
LibroDialogController	Funzionale
GIUSTIFICAZIONE	
La classe è responsabile di un singolo compito, ovvero gestire le interazioni dell'utente nella schermata per aggiungere/modificare un libro. Tutti i metodi e i campi sono strettamente correlati a questo obiettivo: - aggiungiAutore() e rimuoviAutore() gestiscono la lista autori del libro. - conferma() valida i dati inseriti e salva il nuovo libro o le modifiche.	

- setLibro() precompila i campi in modalità modifica. - Gli elementi FXML (fieldTitolo, fieldIsbn, listaAutori, ecc.) sono usati per interagire con i dati immessi dall'utente.

Classe	**COESIONE**
UtenteController	Comunicazionale
GIUSTIFICAZIONE	
La classe gestisce tutte le operazioni relative alla tabella degli utenti, con struttura analoga a LibroController: - inserisciUtente(), modificaUtente(), eliminaUtente() modificano la collezione degli utenti. - cercaUtente() e mostraTuttiUtenti() filtrano/visualizzano i dati. - aggiornaTabella() ricarica e ordina la TableView. - i metodi lavorano sulla stessa struttura informativa (lista utenti) ma svolgono operazioni diverse.	

Classe	**COESIONE**
UtenteDialogController	Funzionale
GIUSTIFICAZIONE	
La classe gestisce un singolo compito: dialog per inserimento/modifica utente. Tutti i metodi e gli elementi dell'interfaccia collaborano per questo obiettivo: - conferma() valida i dati inseriti e crea o aggiorna l'utente. - setUtente() precompila i campi e mostra informazioni sui prestiti attivi (solo in modifica). - La validazione dei campi (nome, cognome, matricola, email) è centralizzata in conferma(). - Gli elementi FXML (campi di testo e label) sono usati per raccogliere e mostrare i dati dell'utente.	

Classe	**COESIONE**
PrestitoController	Comunicazionale
GIUSTIFICAZIONE	
La classe gestisce tutte le operazioni relative ai prestiti e alle statistiche correlate. I metodi lavorano sulla stessa struttura dati (lista prestiti attivi) con funzioni diverse e complementari: - registraPrestito() crea un nuovo prestito con validazioni (disponibilità dell'utente e del libro). - registraRestituzione() chiude un prestito esistente. - aggiornaTabella() ricarica e ordina la vista dei prestiti attivi. - aggiornaStatistiche() calcola totali, ritardi e scadenze imminenti. - La row factory colora le righe in base allo stato (ritardo/scadenza/attivo).	

Classe	**COESIONE**
AlertHelper	Funzionale
GIUSTIFICAZIONE	
La classe è responsabile di un singolo compito: centralizzare la creazione e la visualizzazione degli alert JavaFX. Ogni metodo gestisce un tipo specifico di alert: - mostraErrore() visualizza messaggi di errori. - mostraConferma() mostra conferme di operazioni riuscite. - mostraInfo() gestisce messaggi informativi. - mostraConfermaCancellazione() richiede una conferma esplicita e ritorna un valore booleano. Tutti i metodi sono orientati allo stesso scopo (gestione degli alert), evitando duplicazioni e mantenendo uniforme la comunicazione verso l'utente.	

TABELLE DI ACCOPPIAMENTO

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
MainApp, Biblioteca	Dati
GIUSTIFICAZIONE	
MainApp interagisce con Biblioteca solo attraverso metodi pubblici ben definiti: - Biblioteca.getInstance() per ottenere il singleton. - biblioteca.caricaDati() per caricare i dati all'avvio dell'applicazione. - biblioteca.salvaDati() per salvare i dati alla chiusura. Non accede a dettagli implementativi interni di Biblioteca. Passa solo dati necessari (nessun parametro in questo caso).	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
Biblioteca, Libro	Dati
GIUSTIFICAZIONE	
Biblioteca gestisce una lista di oggetti Libro accedendo solo ai loro metodi pubblici. Le operazioni avvengono sempre tramite l'interfaccia di Libro: - libro.getIsbn(), libro.getTitolo(), libro.isDisponibile() per leggere le informazioni principali. - libro.decrementaCopie() e libro.incrementaCopie() per aggiornare la disponibilità durante prestiti e restituzioni. - libro.getAutori(), libro.getNumeroCopieTotali(), libro.getNumeroCopieDisponibili() quando servono i dettagli del libro ai controller. - Gli oggetti Libro vengono passati ai controller come dati, senza accedere ai campi privati. Biblioteca non accede a campi privati né modifica direttamente lo stato interno dei libri, ma opera sempre attraverso i metodi pubblici.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
Biblioteca, Utente	Dati
GIUSTIFICAZIONE	
Biblioteca gestisce una lista di oggetti Utente e interagiscecon essi solo tramite metodi pubblici. In tutte le operazioni usa esclusivamente l'interfaccia della classe Utente: - utente.getMatricola(), utente.getNome(), utente.getCognome(), utente.getNomeCognome() per leggere le informazioni principali. - utente.haRaggiuntoLimite() per verificare se può effettuare prestiti. - utente.aggiungiPrestito() e utente.rimuoviPrestito() durante la registrazione o chiusura dei prestiti. - Passa gli oggetti Utente ai controller o li restituisce come risultati delle ricerche. Biblioteca non accede mai ai campi privati degli utenti e non ne modifica direttamente lo stato interno: tutto avviene tramite i metodi pubblici.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
Biblioteca, Prestiti	Dati
GIUSTIFICAZIONE	
Biblioteca gestisce una lista di oggetti Prestito e interagisce solo attraverso i loro metodi pubblici: - new Prestito(utente, libro, dataRestituzionePrevista) per creare un nuovo prestito. - prestito.isAttivo() per filtrare prestiti aperti. - prestito.registraRestituzione() per chiudere prestiti.	

- prestito.getDataRestituzioneRevista(), prestito.getUtente(), prestito.getLibro() per ordinamento, filtri e statistiche.
- Passa gli oggetti Prestito ai controller come dati

Biblioteca non accede a campi privati né modifica direttamente lo stato interno dei Prestito.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
Prestito, Libro	Dati
GIUSTIFICAZIONE	

Prestito mantiene un riferimento a un oggetto Libro, ma interagisce solo tramite getter pubblici:

- prestito.getLibro() restituisce il riferimento al libro associato al prestito.
- libro.getTitolo() è usato per formattare la descrizione del prestito (es. in toString() o getStatoDescrizione()).
- libro.getIsbn() può essere richiamato quando serve identificare il libro.

Prestito non modifica mai direttamente lo stato interno del libro (le modifiche alle copie sono gestite esclusivamente da Biblioteca).

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
Prestito, Utente	Dati
GIUSTIFICAZIONE	

Prestito mantiene un riferimento a un oggetto Utente e lo utilizza solo tramite i suoi metodi pubblici:

- prestito.getUtente() restituisce l'utente associato al prestito.
- utente.getNomeCognome() è utilizzato per comporre le stringhe di descrizione.
- Prestito non modifica lo stato di Utente (gestione lista prestiti è responsabilità di Utente stesso).

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
Utente, Prestito	Dati
GIUSTIFICAZIONE	

Utente mantiene una lista di oggetti Prestito (prestiti attivi) e li gestisce solo tramite metodi pubblici della classe:

- aggiungiPrestito(Prestito prestito) aggiunge alla lista un prestito attivo.
- rimuoviPrestito(Prestito prestito) per rimuoverlo quando viene restituito.
- getPrestitiAttivi() restituisce una nuova lista, separata da quella interna, così non può essere modificata dall'esterno.

Utente non accede a metodi interni di Prestito, tratta gli oggetti semplicemente come dati.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
LibroController, Biblioteca	Dati
GIUSTIFICAZIONE	

LibroController interagisce con Biblioteca solo tramite i metodi pubblici messi a disposizione dalla classe:

- biblioteca.getInstance() per ottenere l'istanza del sistema.
- biblioteca.getTuttiLibri() per popolare la tabella iniziale.
- biblioteca.cercaLibri(criterio, tipo) per eseguire ricerche.
- biblioteca.eliminaLibro(isbn) per rimuovere un libro selezionato.

Passa solo parametri necessari (ISBN, criteri di ricerca, o oggetti Libro) e non accede mai ai campi interni di Biblioteca.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
LibroController, LibroDialogController	Controllo
GIUSTIFICAZIONE	
LibroController controlla il flusso del dialog di inserimento/modifica coordinando LibroDialogController: - Crea il dialog con FXMLLoader. - loader.getController() per ottenere l'istanza del dialog. - controller.setLibro(libro) per inizializzare la finestra (libro esistente o nuovo). - dialogStage.showAndWait() per bloccare l'esecuzione finché il dialog non viene chiuso. - controller.isConfermato() per verificare il risultato.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
LibroController. AlertHelper	Dati
GIUSTIFICAZIONE	
LibroController utilizza AlertHelper per mostrare messaggi all'utente, chiamando soltanto i suoi metodi pubblici statici e passando dati semplici (stringhe): - AlertHelper.mostraErrore("Errore", messaggio). - AlertHelper.mostraConferma("Successo", messaggio). - AlertHelper.mostraConfermaCancellazione(oggetto) restituisce un booleano. Il controller non accede a variabili interne, non gestisce alcuno stato dell'altro componente e non c'è dipendenza strutturale: viene solo richiesto un servizio fornito da AlertHelper.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
UtenteController, Biblioteca	Dati
GIUSTIFICAZIONE	
Interazione analoga a LibroController: - biblioteca.getTuttiUtenti() per caricare i dati nella tabella. - biblioteca.cercaUtenti() per effettuare una ricerca. - biblioteca.eliminaUtente() per rimuovere un utente. Passa solo parametri necessari (stringhe, oggetti Utente). Il controller non accede mai a campi interni o dettagli di implementazione della classe Biblioteca.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
UtenteController, UtenteDialogController	Controllo
GIUSTIFICAZIONE	
Il comportamento segue lo stesso schema usato per LibroController–LibroDialogController. UtenteController gestisce il flusso del dialog e decide cosa fare in base all'esito restituito da UtenteDialogController: - Crea il dialog tramite FXMLLoader. - Configura con setUtente(), attende con showAndWait(), verifica con isConfermato(). - UtenteController controlla il flusso in base al risultato booleano.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
UtenteController, AlertHelper	Dati
GIUSTIFICAZIONE	
L'interazione è identica a quella tra **LibroController** e **AlertHelper**. UtenteController utilizza AlertHelper solo per mostrare messaggi all'utente, chiamando i suoi metodi pubblici statici e passando semplici stringhe: - AlertHelper.mostraErrore("Errore", messaggio).	

- AlertHelper.mostraConferma("Successo", messaggio).
- AlertHelper.mostraConfermaCancellazione(oggetto) che restituisce un booleano.

Il controller non accede a variabili interne, non gestisce alcuno stato dell'altro componente e si limita a richiedere i servizi messi a disposizione da AlertHelper.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
PrestitoController, Biblioteca	Dati
GIUSTIFICAZIONE	

PrestitoController interagisce con Biblioteca solo tramite interfaccia pubblica:

- biblioteca.getPrestitiAttivi() per popolare la tabella dei prestiti.
- biblioteca.registraPrestito(matricola, isbn, data) per creare un nuovo prestito.
- biblioteca.registraRestituzione(prestito) per registrare la restituzione.
- biblioteca.getTuttiUtenti() e biblioteca.getTuttiLibri() per riempire le ComboBox del dialog.

Il controller non accede a campi privati né manipola direttamente lo stato interno di Biblioteca, ma si limita a richiederne i servizi.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
PrestitoController, Utente, Libro	Dati
GIUSTIFICAZIONE	

- Non accede a proprietà interne, usa solo il riferimento all'oggetto.
- La ComboBox chiama automaticamente toString() di Utente/Libro per visualizzazione.
- Usa metodi pubblici come utente.haRagguintoLimite() e libro.isDisponibile() per validazioni.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
PrestitoController, AlertHelper	Dati
GIUSTIFICAZIONE	

L'interazione è analoga a quella vista negli altri controller (LibroController e UtenteController). PrestitoController utilizza AlertHelper esclusivamente per mostrare messaggi all'utente, chiamando i metodi pubblici statici della classe e passando soltanto dati semplici (stringhe o testi formattati):

- AlertHelper.mostraErrore("Errore", messaggio).
- AlertHelper.mostraConferma("Successo", messaggio).
- AlertHelper.mostraConfermaCancellazione(oggetto) che restituisce un booleano.

Il controller non accede ad alcuna variabile interna di AlertHelper, non condivide stato e non dipende da dettagli implementativi.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
LibroDIalogController, Biblioteca	Dati
GIUSTIFICAZIONE	

LibroDialogController interagisce con Biblioteca solo per salvare il libro:

- biblioteca.aggiungiLibro(nuovoLibro) per nuovi libri.
- Biblioteca.modificaLibro(libroCorrente) per modifiche.

Passa solo oggetti Libro, non accede a dettagli implementativi.

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
UtenteDialogController, Biblioteca	Dati
GIUSTIFICAZIONE	

Interazione analoga a LibroDialogController:

- Biblioteca.aggiungiUtente(nuovoUtente) per nuovi utenti.

- Biblioteca.modificaUtente(utenteCorrente) per modifiche.

Passa solo oggetti Utente come parametro e non accede a campi privati né a dettagli interni di Biblioteca.