# Analisi Architetturale: Coesione e Accoppiamento

Gruppo 12

8 dicembre 2025

## 1 Analisi delle Componenti

Di seguito viene riportata l'analisi dettagliata delle metriche di qualità software (Coesione e Accoppiamento) per le principali classi del sistema.

### 1.1 Classi Entità: Utente / Libro

- **Coesione:** Funzionale.
  Le classi rappresentano fedelmente le entità del dominio; ogni metodo è strettamente correlato alla gestione dei dati intrinseci dell'entità.
- **Accoppiamento:** Per Dati.
  L'interazione con l'esterno avviene scambiando esclusivamente tipi primitivi o dati semplici, garantendo la massima indipendenza.

### 1.2 Classe Entità: Prestito

- **Coesione:** Funzionale.
  Tutti i metodi concorrono a definire l'entità *Prestito* e le relative operazioni di gestione stato.
- **Accoppiamento:** Di Timbro (Stamp Coupling).
  A differenza delle entità base, i metodi di questa classe richiedono il passaggio di strutture dati composite (riferimenti a oggetti `Libro` o `Utente`) piuttosto che soli tipi primitivi (es. metodo `setLibro`).

### 1.3 Classi Gestore: Libro / Utente / Prestito

- **Coesione:** Funzionale.
  Ogni metodo persegue un unico obiettivo specifico, come l'aggiunta di un elemento alla collezione o la persistenza dei dati su file.
- **Accoppiamento:** Di Controllo.
  Il metodo `ordinaLista` richiede in input un oggetto `Comparator`. Questo parametro esterno altera la logica interna di ordinamento della struttura dati visualizzata.

## 1.4 Handler: UtenteHandler / LibroHandler / PrestitoHandler

- **Coesione:** Sequenziale.
  I metodi `onAdd` e `onEdit` orchestrano una serie di operazioni che devono essere eseguite in un ordine rigoroso (startup del controller e inizializzazione del form) affinché il flusso termini correttamente.

- **Accoppiamento:** Di Controllo.
  Il metodo `ordina` accetta come parametro una stringa che determina l'algoritmo di ordinamento da applicare, influenzando direttamente il flusso di esecuzione del programma.

## 1.5 Classe: PrincipaleController

- **Coesione:** Temporale.
  Il metodo `initialize` raggruppa operazioni eterogenee la cui unica relazione è il momento di esecuzione, ovvero la fase di avvio dell'applicazione e dei componenti grafici.

- **Accoppiamento:** Di Timbro.
  La maggioranza dei metodi (es. `setArea`, `ordinaTabella`) opera passando o ricevendo intere strutture dati o oggetti complessi, invece di singoli valori.

## 1.6 Controller Form: FormLibroController / FormUtenteController

- **Coesione:** Comunicazionale.
  I metodi operano sullo stesso insieme di dati (i campi del form e l'oggetto in modifica) per eseguire azioni distinte ma correlate (validazione, salvataggio, annullamento).

- **Accoppiamento:** Di Timbro.
  *Timbro:* Le classi ricevono l'intera istanza dell'oggetto per popolare i campi.

## 1.7 Controller Form: FormPrestitoController

- **Coesione:** Comunicazionale.
  Simile agli altri form, ma con una complessità maggiore: opera sull'aggregazione di dati provenienti da fonti diverse (selezione Utente, selezione Libro, date) per produrre l'output finale (il Prestito).

- **Accoppiamento:** Di Timbro (Elevato).
  Presenta un accoppiamento di timbro più marcato rispetto agli altri form, poiché deve gestire e collegare istanze multiple di oggetti complessi (`Libro`, `Utente` e `Prestito`) per garantire la coerenza del vincolo relazionale.

## 1.8 Classe: libraryManagerApp

- **Coesione:** Funzionale.
  È la classe main dell'applicazione. Ogni suo metodo ha il solo e unico scopo di caricare la scena principale.

- **Accoppiamento:** Di Timbro.
  Il metodo `start` richiede uno `Stage` passato come parametro. C'è una chiara dipendenza con il modulo **Stage** ma d'altro canto è la classe main di un programma in JavaFX quindi è normale che sia presente.