Bibliosoft – Commenti ai diagrammi

Indice

1 Diagramma delle Classi (Parte 1): modelli

Il pacchetto `modelli` contiene `Utente`, `Libro`, `Prestito` e `StatoPrestito`.

1.1 Coesione

- `Utente` raggruppa i dati dell'utente e le operazioni sui `prestitiAttivi` tramite `aggiungiPrestito`, `rimuoviPrestito` e `haPrestitiAttivi`, mantenendo queste funzionalità nello stesso punto.
- `Libro` raccoglie le informazioni del libro e la gestione delle copie con `isDisponibile` e `contieneAutore`, con metodi coerenti con i suoi attributi.
- `Prestito` rappresenta l'associazione tra `Utente` e `Libro` e gestisce le date e lo stato tramite `aggiornaStato`.
- `StatoPrestito` isola i valori `IN_CORSO`, `IN_RITARDO`, `CONCLUSO` come insieme chiuso di stati del prestito.

1.2 Accoppiamento

- `Prestito` dipende direttamente da `Utente` e `Libro`, e `Utente` mantiene una lista di `Prestito`. L'accoppiamento è naturale per rappresentare il dominio applicativo.
- L'implementazione di `Comparable` su `Utente`, `Libro` e `Prestito` introduce una dipendenza minima verso l'ordinamento, utile per la gestione delle liste.

1.3 Scelte di buona progettazione

- `matricola` e `isbn` indicati come **final** chiariscono l'identità stabile degli oggetti.
- La separazione di `StatoPrestito` come enumerazione evita valori inconsistenti nello stato del prestito.

2 Diagramma delle Classi (Parte 2): servizi, archivi, strumenti

Sono presenti `Archivio`, `Sottoarchivio<T>`, `InterfacciaSottoarchivio<T>`, `ServizioLibri`, `ServizioUtenti`, `ServizioPrestiti`, `ServizioArchivio` e `Validatore`.

2.1 Coesione

- `InterfacciaSottoarchivio<T>` e `Sottoarchivio<T>` raggruppano operazioni standard (`aggiungi`, `modifica`, `rimuovi`, `lista`, `cerca`, `conta`) su collezioni omogenee: coesione alta.
- `Archivio` mantiene insieme i sottoarchivi di `libri`, `utenti` e `prestiti`, offrendo metodi specifici per ciascun tipo.

- `ServizioLibri`, `ServizioUtenti` e `ServizioPrestiti` sono focalizzati sulle operazioni applicative delle rispettive aree.
- `ServizioArchivio` concentra le responsabilità di `carica()` e `salva()`.
- `Validatore` isola la validazione di `Libro` e `Utente` dalle operazioni di gestione.

2.2 Accoppiamento

- I `Servizi` dipendono da `Archivio` invece che dai singoli sottoarchivi, semplificando la rete di dipendenze.
- L'uso di `cerca(InterfacciaFiltro<...> filtro)` riduce la dipendenza tra criteri di ricerca e struttura di memorizzazione.

2.3 Scelte di buona progettazione

- La persistenza è confinata in `ServizioArchivio`.
- La logica di prestito è concentrata in `ServizioPrestiti` con operazioni come `registraPrestito`, `registraRestituzione` e `aggiornaRitardi`.

3 Diagramma delle Classi (Parte 3): controller

Il pacchetto `controller` include `Bibliosoft`, `ControllerPrincipale`, `ControllerDashboard`, `ControllerLibri`, `ControllerUtenti` e `ControllerPrestiti`.

3.1 Coesione

- `ControllerLibri` è coeso su operazioni della tabella e dei dialog del libro: `inizializzaTabella`, `onRicerca`, `onAggiungi`, `onModifica`, `onElimina`, `creaDialogLibro`.
- `ControllerUtenti` replica lo stesso stile per `Utente` con `inizializzaTabella`, `onRicerca`, `onAggiungi`, `onModifica`, `onElimina`, `creaDialogUtente`.
- `ControllerPrestiti` concentra le azioni di registrazione e gestione della vista prestiti (`onRegistraPrestito`, `onRegistraRestituzione`, `aggiorna`).
- `ControllerDashboard` è focalizzato su `impostaServizi` e sui metodi di riepilogo.

3.2 Accoppiamento

- `ControllerPrincipale` è il punto di maggior accoppiamento: possiede riferimenti a più servizi (`ServizioLibri`, `ServizioUtenti`, `ServizioPrestiti`, `ServizioArchivio`) e ai controller di area.
- Questa scelta mantiene però separati i controller specifici, evitando dipendenze dirette tra `ControllerLibri`, `ControllerUtenti` e `ControllerPrestiti`.

3.3 Scelte di buona progettazione

- I metodi di navigazione e aggiornamento nel `ControllerPrincipale` (`cambiaTab`, `caricaVistaLibri`, `caricaVistaUtenti`, `caricaVistaPrestiti`, `caricaVistaDashboard`) rispecchiano la struttura dell'applicazione.
- Le azioni globali `onSalvaArchivio` e `chiudiApplicazione` raggruppano operazioni trasversali in un punto unico.

4 Diagramma delle Classi (Parte 4): filtri

Il pacchetto `filtri` include `InterfacciaFiltro<T>`, `FiltroLibro`, `FiltroUtente` e `FiltroPrestito`.

4.1 Coesione

- `FiltroLibro` è centrato su ricerche come `ricerca` e `ricercaIsbn`.
- `FiltroUtente` concentra `ricerca`, `ricercaMatricola` e `ricercaUtentiAttivi`.
- `FiltroPrestito` concentra filtri legati allo stato e alle ricerche per `matricola`, in linea con `StatoPrestito`.

4.2 Accoppiamento

- I filtri forniscono oggetti `InterfacciaFiltro<...>` utilizzati da `cerca(...)`: la logica dei criteri resta separata dai dettagli di `Archivio` e `Sottoarchivio`.

5 Diagrammi di sequenza (UC1–UC17)

I diagrammi mostrano un flusso ricorrente con attori e componenti: `Bibliotecario` → `Vista (GUI)` o `VistaLibri (GUI)` o `VistaUtenti (GUI)` o `VistaPrestiti (GUI)` → `ControllerPrincipale` o controller di area → `Servizi`.

5.1 Coesione osservata nei flussi

- UC3–UC6 confermano la coesione di `ControllerLibri`: le azioni della vista richiamano `onAggiungi`, `creaDialogLibro(null)`, `onModifica`, `onElimina`, `onRicerca`.
- UC8–UC11 confermano la coesione di `ControllerUtenti` con lo stesso schema operativo e l'uso di `creaDialogUtente(null)`.
- UC14–UC15 sono centrati su `ControllerPrestiti` e sulle operazioni `onRegistraPrestito` e `onRegistraRestituzione`.
- UC1, UC2, UC7 e UC12 mostrano l'uso di `cambiaTab()` e `aggiorna()`, coerenti con il ruolo di coordinamento del `ControllerPrincipale`.

5.2 Accoppiamento osservato nei flussi

- Le viste interagiscono con i controller e non direttamente con `ServizioLibri`, `ServizioUtenti` o `ServizioPrestiti`, mantenendo separata la GUI dalla logica applicativa.
- `ControllerPrincipale` appare come snodo di molte interazioni, coerente con la sua funzione, ma con un accoppiamento più elevato rispetto ai controller di area.

5.3 Gestione errori e chiusura

- UC16 evidenzia un flusso alternativo con errore I/O durante `onSalvaArchivio(...)`: la presenza di `ServizioArchivio` e di `mostraErrore(String messaggio)` nei controller è coerente con questa gestione.
- UC17 mostra la chiusura tramite `chiudiApplicazione(...)` con distinzione tra presenza o assenza di modifiche pendenti, confermando il ruolo centrale del `ControllerPrincipale`

6 Sintesi

Nel complesso, i diagrammi mostrano una struttura coerente: `controller` → `servizi` → `archivi` → `modelli`, con le ricerche mediate dai `filtri`. La coesione è alta nei controller di area e nei servizi specifici, mentre l'accoppiamento più significativo si concentra nel `ControllerPrincipale`, in linea con i metodi di navigazione e con le operazioni globali di `carica`, `salva` e chiusura dell'applicazione.