Corso di Programmazione ad Oggetti – a.a. 2013/2014
Corso di laurea in Ingegneria e Scienza Informatiche – Università di Bologna

Relazione dello sviluppo dell'applicazione “SAW – Sit And Watch: sistema per la gestione di un cinema multisala”

Componenti del gruppo: Cristina Gentilini, Marina Londei

1 Analisi del problema

È stata realizzata un' applicazione per la gestione di un cinema multisala. Vengono gestite le richieste di due tipi di utenza: (i) proprietario e servizi annessi, (ii) cassiere e servizi annessi.

Nello specifico, la parte riguardante l'utente “proprietario” consente di:

- Vedere il riepilogo di tutti i film e le relative visioni in programmazione divise per sale.
- Aggiungere alla programmazione una nuova proiezione di un film.
- Modificare le proiezioni in precedenza inserite.
- Eliminare una proiezione.

La parte riguardante l'utente “dipendente” consente invece di:

- Visualizzare l’elenco dei film in programmazione nelle diverse sala.
- Scelto il film, avere la possibilità di selezionare la proiezione che desidera.
- Prenotare il numero di posti (pari al massimo al numero di posti liberi rimasti) che desidera.

In base alla tipologia di accesso che è stato eseguito, proprietario o dipendente, dipenderà la GUI che sarà mostrata dall’applicazione: in questo modo l’utenza avrà a disposizione solo i dati di proprio interesse.

L’applicazione dovrà garantire la persistenza delle informazioni inserite dai vari utilizzatori memorizzando e caricando i dati ad ogni chiusura e ad ogni lancio dell’applicazione stessa.

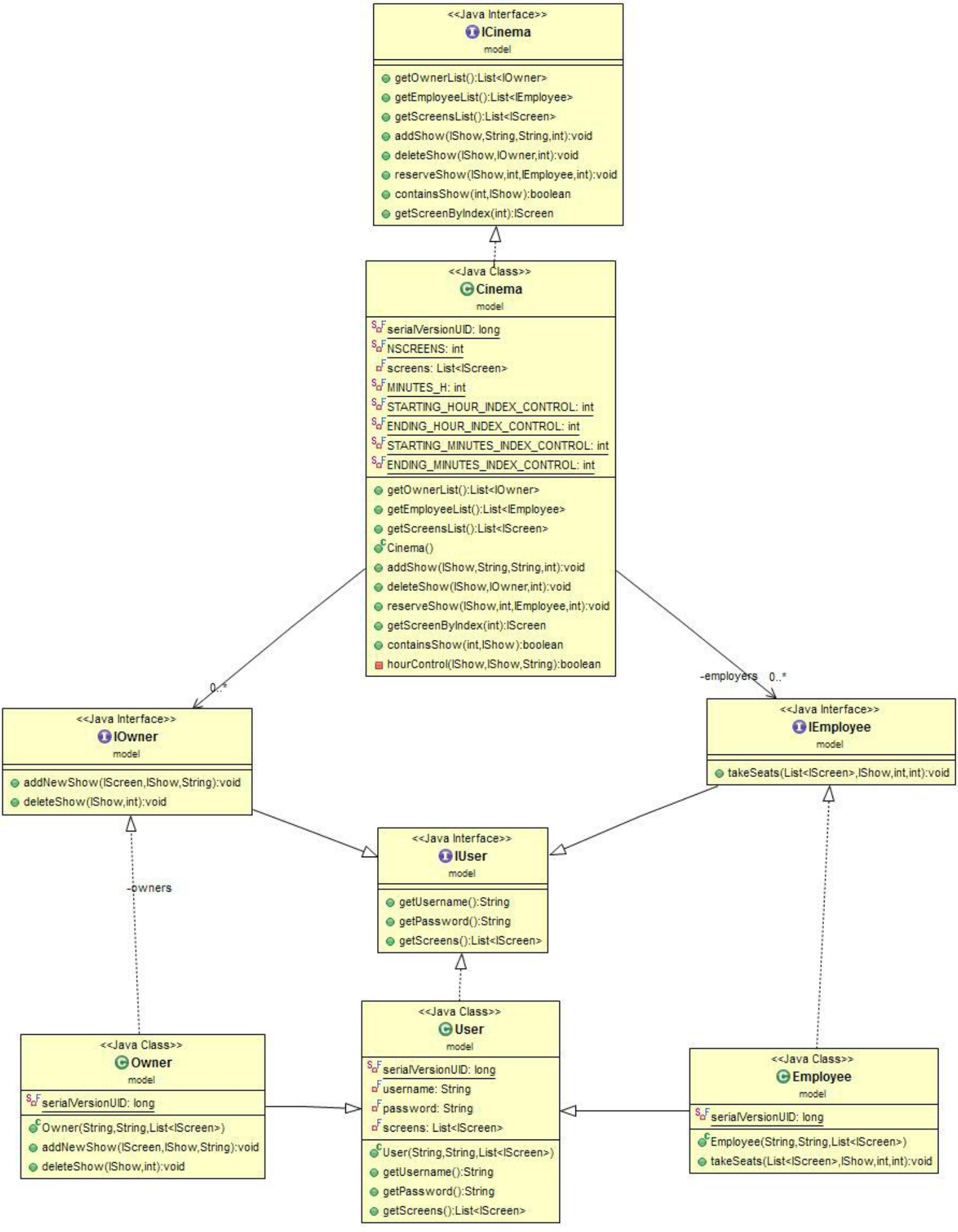
2 Progettazione architetturale

In questa fase si è valutato, in base alle classi e al tipo di applicazione che si voleva fornire, le modalità e i pattern architetturali più prestanti ed efficaci.

Per l'intera fase di progettazione è stato utilizzato il pattern MVC (Model – View – Controller) per garantire la netta separazione tra modellazione e visualizzazione. In questo modo si è riusciti a dividere il lavoro in due parti distinte e indipendenti, e solo in seguito è stato eseguito il merge.

Segue ora la presentazione degli aspetti più rilevanti relativi all'architettura del sistema.

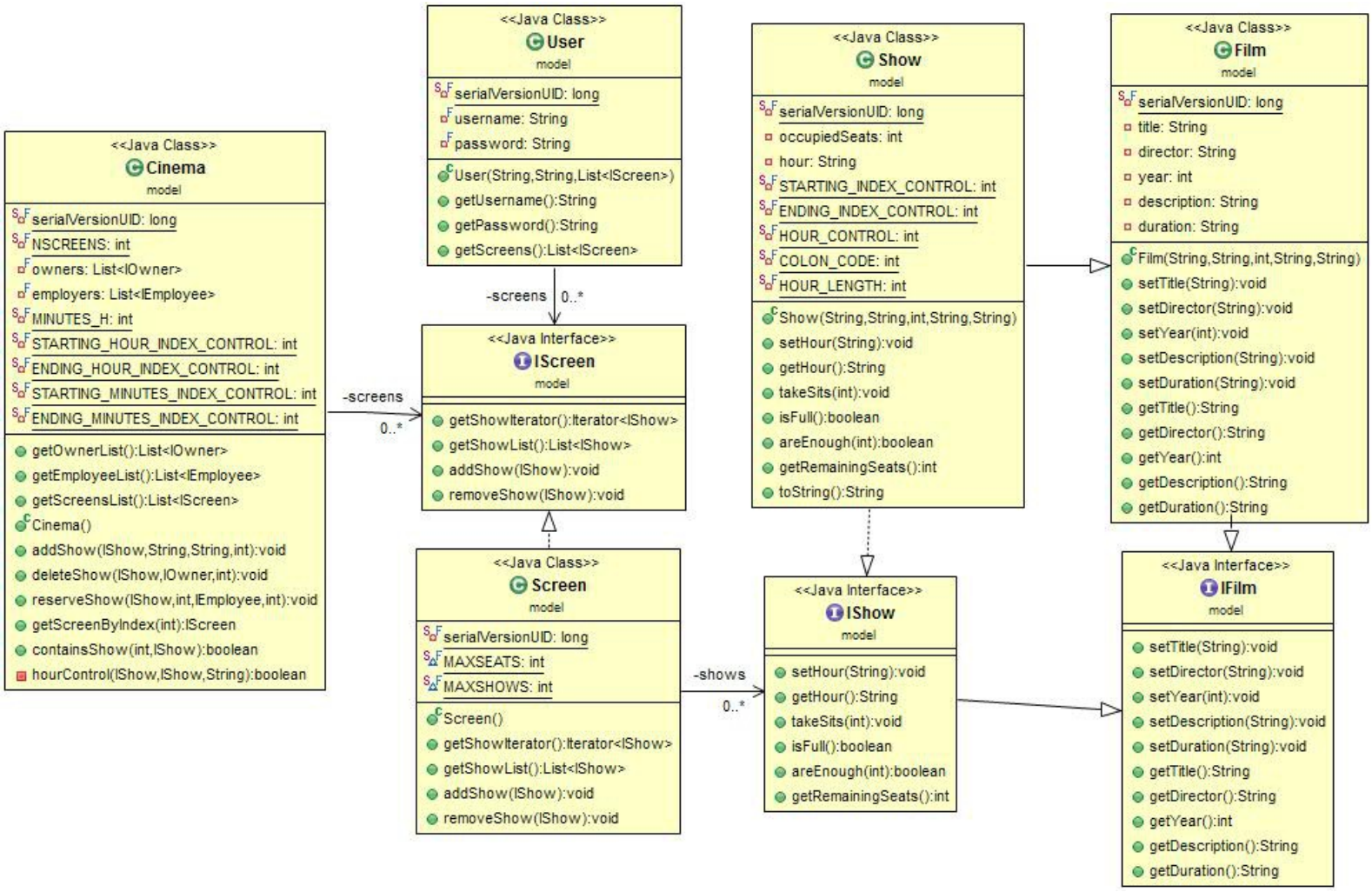
Diagramma UML relativo alla parte del model (per quanto riguarda l'interazione degli utenti con esso).



Descrizione degli aspetti principali:

- ICinema e Cinema: rappresentano rispettivamente l'interfaccia e l'implementazione del modello. Quest'ultima implementa l'interfaccia Serializable per garantire la persistenza dei dati.
- IUser: interfaccia che modella un generico utente (proprietario o commesso) che è in grado di fare il login all'applicazione.
- Owner-Employee: classi che modellano un generico proprietario e un generico commesso.

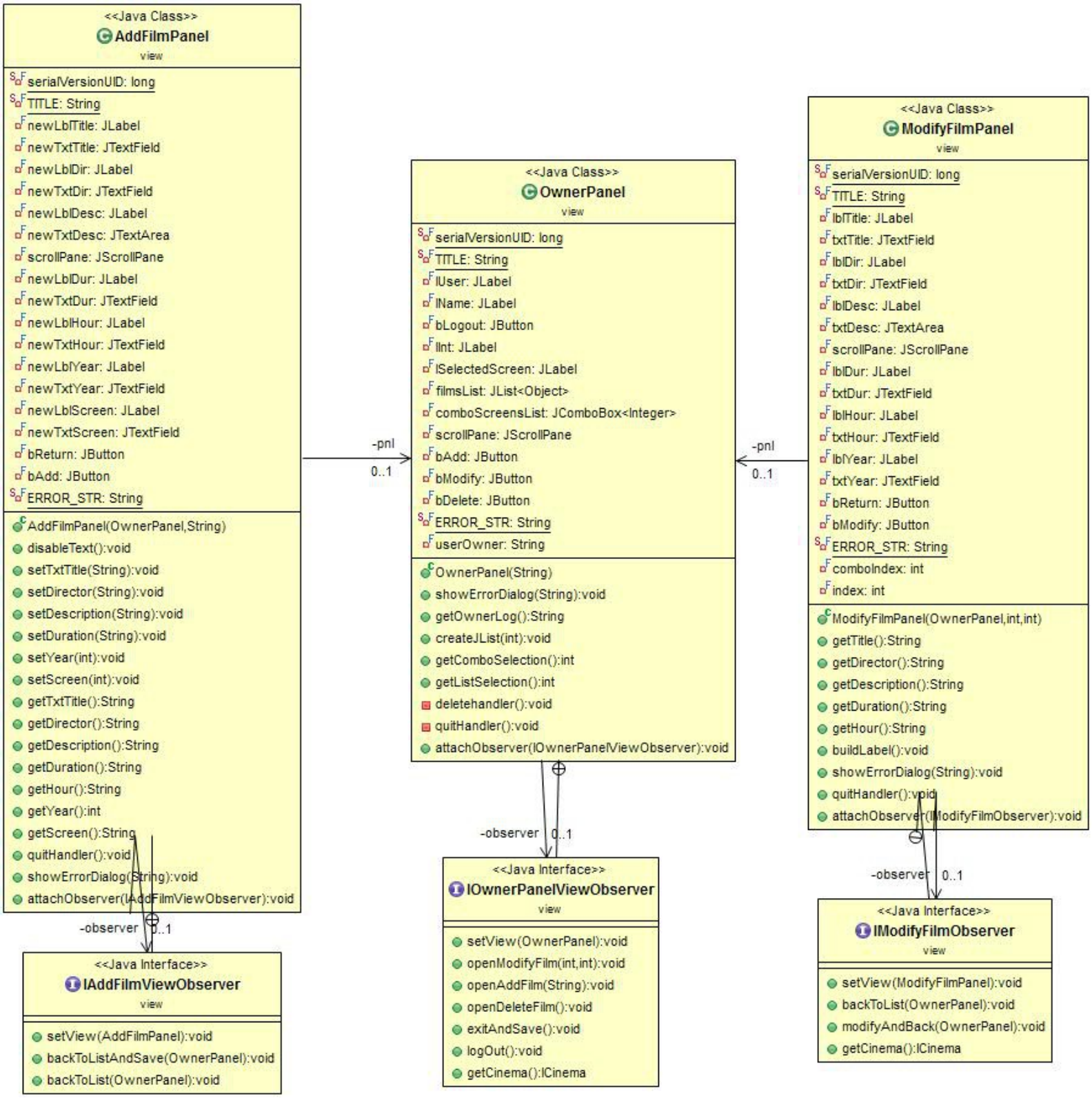
Diagramma UML relativo alla parte del Model (per quanto riguarda le associazioni dei film e degli show con esso).



Descrizione degli aspetti principali:

- IScreen e Screen : interfaccia e implementazione di una sala. Quest'ultima implementa Serializable.
- IShow e Show : interfaccia e implementazione che descrivono un film in proiezione.
- IFilm e Film: interfaccia e implementazione di un film.

Diagramma UML relativo alla parte della view (riguardante la parte del proprietario).

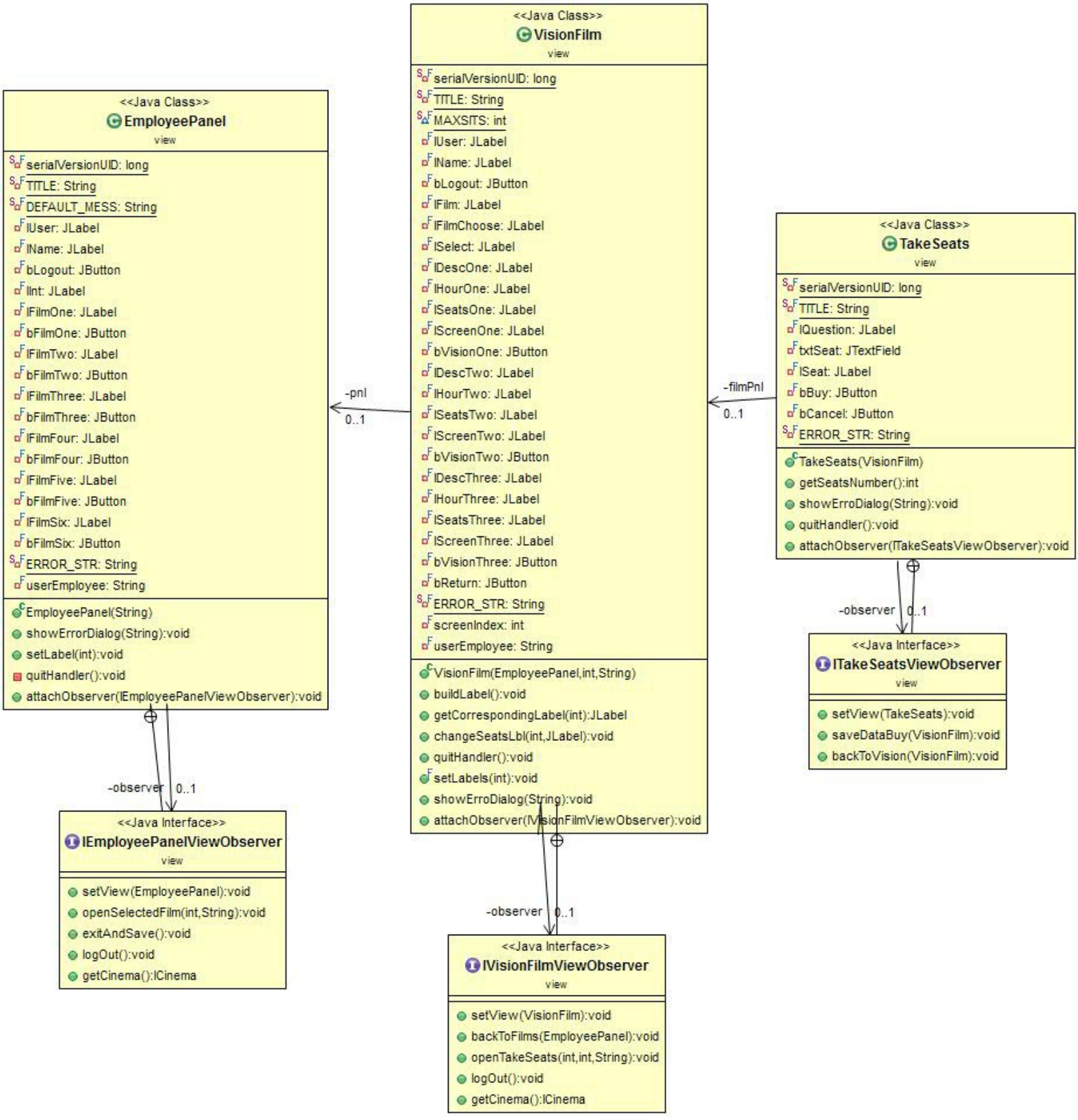


Descrizione degli aspetti principali:

- OwnerPanel e IOwnerPanelViewObserver: GUI e relativa interfaccia per l'observer della vista del proprietario.
- AddFilmPanel e IAddFilmViewObserver: GUI e relativa interfaccia per l'observer della vista dell'aggiunta di un film. Operazione gestita dal proprietario.

- ModifyPanel e IModifyFilmObserver: GUI e relativa interfaccia per l'observer della vista della modifica di un film presente nella lista. Operazione gestita dal proprietario.

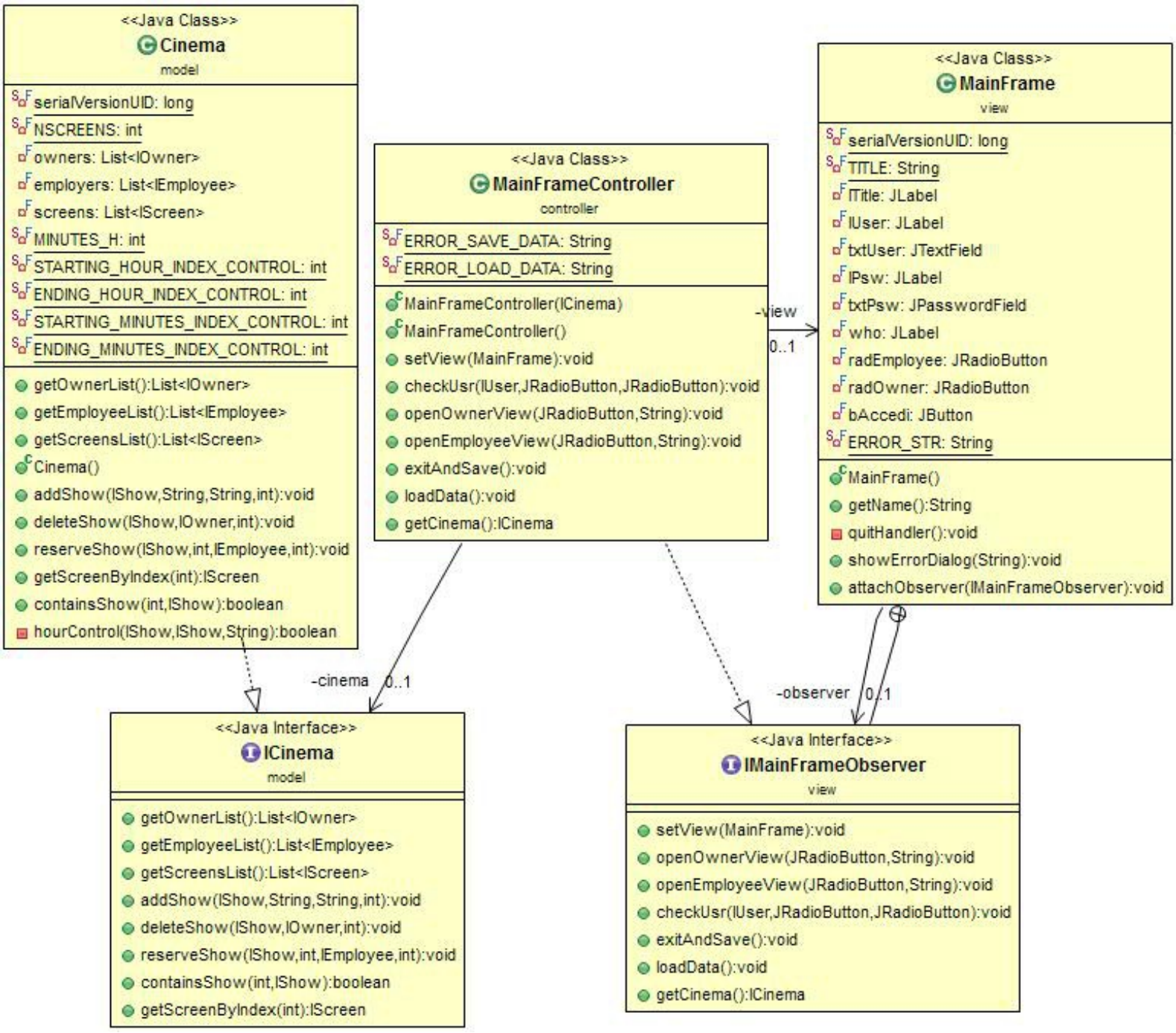
Diagramma UML relativo alla parte della view (riguardante la parte del dipendente).



Descrizione degli aspetti principali:

- EmployeePanel e IEmployeePanelViewObserver: GUI e relativa interfaccia per l'observer della vista del dipendente.
- VisionPanel e IVisionPanelViewObserver: GUI e relativa interfaccia per l'observer della vista del riepilogo delle proiezioni di un film. Gestito dal dipendente.
- TakeSeats e ITakeSeatsViewObserver: GUI e relativa interfaccia per l'observer della vista dell'acquisto dei biglietti. Gestito dal dipendente.

Diagramma UML relativo all'interazione tra la parte di view, model e controller dell'applicazione, in particolare riguardo il frame del login e il suo controller.



Descrizione degli aspetti principali:

- Il diagramma UML rappresenta l'interazione tra le classi dovuta all'utilizzo del pattern MVC.
- Ogni classe della view presenta un'interfaccia observer che racchiude i metodi per gestire eventi provenienti dalla view e modificare di conseguenza i dati del modello.
- Questa interfaccia viene implementata da un controller apposito, che si occupa effettivamente di gestire i cambiamenti sia nella view che nel model, aggiornando entrambi al momento delle modifiche richieste dall'utente. In questo modo, l'unica parte ad avere un riferimento al model è il controller.

3 Organizzazione dei Package

Viene ora effettuata una analisi dell'organizzazione in package dell'applicazione.

- **controller:** contiene i sorgenti che incapsulano il comportamento dei controller relativi alla MainFrame e alle view, sia per il dipendente che per il proprietario. Le classi più importanti contenute in questo package sono:
  - MainFrameController
  - EmployeePanelController
  - OwnerPanelController

- **exceptions:** contiene l'insieme delle classi relative a tutte le possibili eccezioni che possono verificarsi nell'applicazione.

- **main:** contiene solamente la classe Main, il main dell'applicazione

- **model:** contiene i sorgenti necessari per l'implementazione della parte del modello dell'applicazione.

- **view:** contiene i sorgenti del codice usato per implementare tutta la GUI dell'applicazione.

4 Suddivisione del lavoro

Per la realizzazione del progetto il lavoro tra i vari componenti è stato suddiviso come segue:

- Cristina Gentilini si è occupata dello sviluppo della view e dei controller (e relativa gestione) per quanto riguarda la parte del dipendente.
- Marina Londei si è occupata della realizzazione del modello, delle eccezioni e dei controller (e relativa gestione) per quanto riguarda la parte del proprietario.

Alcune delle parti del progetto sono state realizzate in sinergia:

- Diagrammi UML e le relative descrizioni di tali diagrammi.
- Definizione dell'interfaccia per l'observer del frame principale dell'applicazione e implementazione del relativo controller.
- Classe Main dell'applicazione.
- Correzione finale del codice e di errori di CheckStyle e PMD.

5.1 Progettazione di dettaglio: parte di Cristina Gentilini

Questa parte di relazione è di responsabilità del solo componente del gruppo **Cristina Gentilini**

- Descrizione dettagliata della "View" (classe Spring Layout): la view rappresenta la parte della GUI dell'applicazione.
    - AddFilmPanel: la classe è utilizzata nella parte del proprietario, richiamata quando l'utente richiede di aggiungere una nuova visione di un film alle programmazioni già esistenti.
    - EmployeePanel: questa classe è la prima che è mostrata quando si accede come dipendente. Mostra l'elenco di tutti film in programmazione in quel momento.
    - MainFrame: quando si manda in esecuzione l'applicazione, questa è la prima classe che è richiamata. All'interno di questa è eseguito il login, distinguendo se si voglia accedere come proprietario o come dipendente.
    - ModifyFilmPanel: la classe è utilizzata nella parte del proprietario, è richiamata quando l'utente vuole modificare una visione inserita in precedenza.
    - OwnerPanel: questa classe è la prima che è mostrata quando si accede come proprietario. Mostra l'elenco di tutti film in programmazione suddivisi per sale. La classe contiene tre JButton: uno apre il pannello che permette di aggiungere una nuova visione, uno permette di modificare la visione selezionata tra le preesistenti e uno elimina la visione selezionata tra quelle presenti nell'elenco.
    - TakeSeats: la classe che effettivamente permette di inserire in un JTextField il numero di posti che si vogliono acquistare. Questa classe è richiamata da VisionFilm quando si sceglie l'orario della visione a cui si vuole assistere.
    - VisionFilm: il dipendente che accede seleziona il film di cui desidera acquistare i biglietti e viene aperta questa classe, nella quale vengono elencati gli orari delle visioni possibili.

- Implementazione dei controller riguardanti la parte del dipendente.
    - EmployeePanelController: implementa l'interfaccia IEmployeePanelViewObserver, selezionato in film che si desidera vedere apre la view VisionPanel, passandogli le informazioni necessarie per riconoscere il film.
    - VisionFilmController: implementa l'interfaccia IVisionFilmViewObserver. Permette di tornare alla view precedente se non si vogliono acquistare biglietti, altrimenti apre, attraverso il metodo openTakeSeats, la classe TakeSeats.
    - TakeSeatsController: implementa ITakeSeatsViewObserver. Con il metodo saveDataBuy acquisisce da input il numero di biglietti che si vuole acquistare e salva la modifica.

5.2 Progettazione di dettaglio: parte di Marina Londei

La seguente parte di progettazione è a cura del membro del gruppo **Londei Marina.**

- Descrizione dettagliata del "Model" (MVC pattern): il modello rappresenta la gestione e organizzazione dei dati da parte dell'applicazione.
    - User: rappresenta la classe in cui viene modellato un generico utente che utilizza l'applicazione, dotato di username, password e lista di schermi sui quali gestire le proiezioni

e le prenotazioni degli spettacoli. Questa classe viene estesa per modellare nel dettaglio due tipi di utente: proprietario e commesso.

- Owner: estende la classe User, e rappresenta la modellazione dell'utente di tipo “Proprietario”, con le sue specifiche funzioni e permessi sugli spettacoli. Questo tipo di utente può aggiungere o cancellare proiezioni. La modifica viene invece gestita a livello di view.
- Employee: estende anch'esso la classe User, e rappresenta la modellazione dell'utente di tipo “Commesso”, con le sue specifiche funzioni e permessi sugli spettacoli. Questo tipo di utente si occupa di prenotare i posti richiesti per specifici spettacoli.
- Film: è la classe base che descrive la struttura di uno spettacolo. Si compone di titolo, descrizione, regista, anno e durata. Da questa semplice classe si evolve e si specifica l'intero modello.
- Show: è la classe che estende film, e rappresenta l'oggetto “spettacolo”. La sua caratteristica è di possedere, oltre che i campi ereditati dalla classe Film, l'orario come informazione aggiuntiva: ciò viene utilizzato per distinguere le diverse proiezioni di uno stesso film. Possiede inoltre le informazioni riguardanti il numero di posti occupati per una determinata proiezione.
- Screen: è la classe che rappresenta la singola sala, nella quale è possibile aggiungere o rimuovere una proiezione (tutte gestite tramite una lista linkata). Possiede un numero massimo di posti e di proiezioni giornaliere.
- Cinema: rappresenta il modello vero e proprio e la sua gestione. Possedendo una lista linkata di sale, può agire sulle singole e da lì sulle singole proiezioni, fornendo metodi di gestione sia per l'utente “proprietario” che per l'utente “commesso”. Possiede inoltre due liste in cui vengono memorizzati proprietari e commessi che fanno parte del cinema. Questi vengono istanziati all'interno del costruttore di questa classe, e aggiunti alle rispettive liste. “Cinema” offre metodi per aggiungere e cancellare proiezioni (utilizzati dal proprietario), e per acquistare posti (utilizzato dal commesso). Nel metodo per l'aggiunta di un nuovo spettacolo, viene fatto un controllo sull'orario di proiezione gestito tramite un metodo privato: esso si occupa di controllare che le diverse proiezioni non si sovrappongano (effettuando un controllo sulla durata dei singoli spettacoli).

Note: la classe “Show” presenta l'implementazione del metodo toString, utilizzato per stampare le informazioni riguardanti lo spettacolo inserito all'interno della JList presente nella view.

- Implementazione dei controller riguardanti la parte del proprietario.
  - AddFilmPanelController: implementa l’interfaccia IAddFilmViewObserver. Permette di aggiungere una nuova proiezione in sala (se non è già stato raggiunto il numero massimo di proiezioni) effettuando un controllo: se è già presente un film nella sala selezionata, l’unica cosa che è possibile cambiare è l’orario di proiezione; se, in caso contrario, non c’è alcun film ancora in proiezione, viene aperto un pannello con tutti campi scrivibili.
  - ModifyFilmController: implementa l’interfaccia IModifyFilmObserver. Permette di modificare il film selezionato.
  - OwnerPanelController: implementa l’interfaccia IOwnerPanelViewObserver. Apre i diversi panel a seconda della scelta effettuata, se di modifica, aggiunta, o eliminazione di una proiezione (quest’ultima gestita soltanto a livello di controller).

5.3 Progettazione di dettaglio: parte di Gentilini Cristina e Londei Marina

In comune è stata realizzata la classe MainFrameController che si occupa di gestire il login degli utenti, verificando la validità dell'username e della password inseriti.
Inoltre, a progettazione ultimata, è stata svolta una verifica degli errori di CheckStyle e PMD ed eventuali modifiche al codice per renderlo più leggibile.
Come già specificato, anche la realizzazione dei diagrammi UML è stata fatta in comune.

6 Note finali

- **Note** riguardanti l'uso dell'applicazione:
    - La lista dei proprietari e dipendenti è già stata in precedenza inserita in una lista, quindi non è modificabile, ne è possibile aggiungerne altri al momento del login.
      La lista viene istanziata al momento della creazione del cinema, all'interno del costruttore.

      Di seguito, la **lista di username e password dei proprietari** per accedere e utilizzare l' applicazione:

Username	Password
Owner1	abcde
Owner2	oop13

      Di seguito, la **lista di username e password dei dipendenti** per accedere e utilizzare l'applicazione:

Username	Password
Employee1	uxywz
Employee2	56789
Employee3	sawsw

- **Note** riguardanti le modalità di lavoro:
    - In fase di progettazione, sono stati riassegnati i ruoli per ottimizzare la divisione tra i componenti del gruppo. Si è preferito adattare la divisione del lavoro al modello utilizzato, scegliendo di affidare view e model a membri diversi, così da lavorare in contemporanea.
    - Anche il controller è stato diviso tra due parti indipendenti, in modo che l'ammontare di ore svolte in comune fosse inferiore al 30%.