Corso di laurea in Ingegneria e Scienze Informatiche – Università di Bologna

Corso di Programmazione ad Oggetti – a.a. 2013/2014

Relazione sullo sviluppo dell'applicazione gestionale "MyMuseo – Ticket Booking: gestione della prenotazione di biglietti per le mostre"

Il gruppo è formato da: Conti Laura, Tassinari Francesca

1) Analisi del problema

L'applicazione realizzata ha lo scopo di amministrare le attività riguardanti la gestione di mostre all'interno di un museo. Un altro scopo dell'applicazione è quello di avvicinare più persone al mondo dei musei, incentivando l'uso dell'applicazione stessa per ottenere uno sconto sul prezzo del biglietto di ingresso. Sono gestite le richieste di due diversi tipi di utenti: (i) gli amministratori con i servizi a loro connessi, (ii) i visitatori con i servizi a loro connessi.

In particolare, la parte che riguarda il tipo di utente "amministratore" permette le seguenti funzionalità:

- Al login, consente di visionare il riepilogo delle mostre in programma.
- Aggiungere alle mostre in programma una nuova mostra.
- Modificare i dati delle mostre inserite in precedenza.
- Eliminare una mostra dal programma.

Per quanto riguarda il tipo di utente "visitatore", l'applicazione permette di:

- Al login, visionare il programma delle mostre.
- Una volta scelta la mostra desiderata, prenotare un biglietto allo scopo di ottenere uno sconto.

La tipologia di utente deve essere selezionata all'atto del login e, in base alla tipologia scelta tra amministratore e visitatore, sarà in seguito mostrata una GUI specifica. Questo per fare in modo che ogni utente visualizzi soltanto le funzionalità a lui connesse.

L'applicazione viene concepita in modo tale da assicurare che le informazioni che sono state inserite siano persistenti. Pertanto i dati saranno caricati ad ogni apertura e memorizzati ad ogni chiusura dell'applicazione.

2) Progettazione architetturale

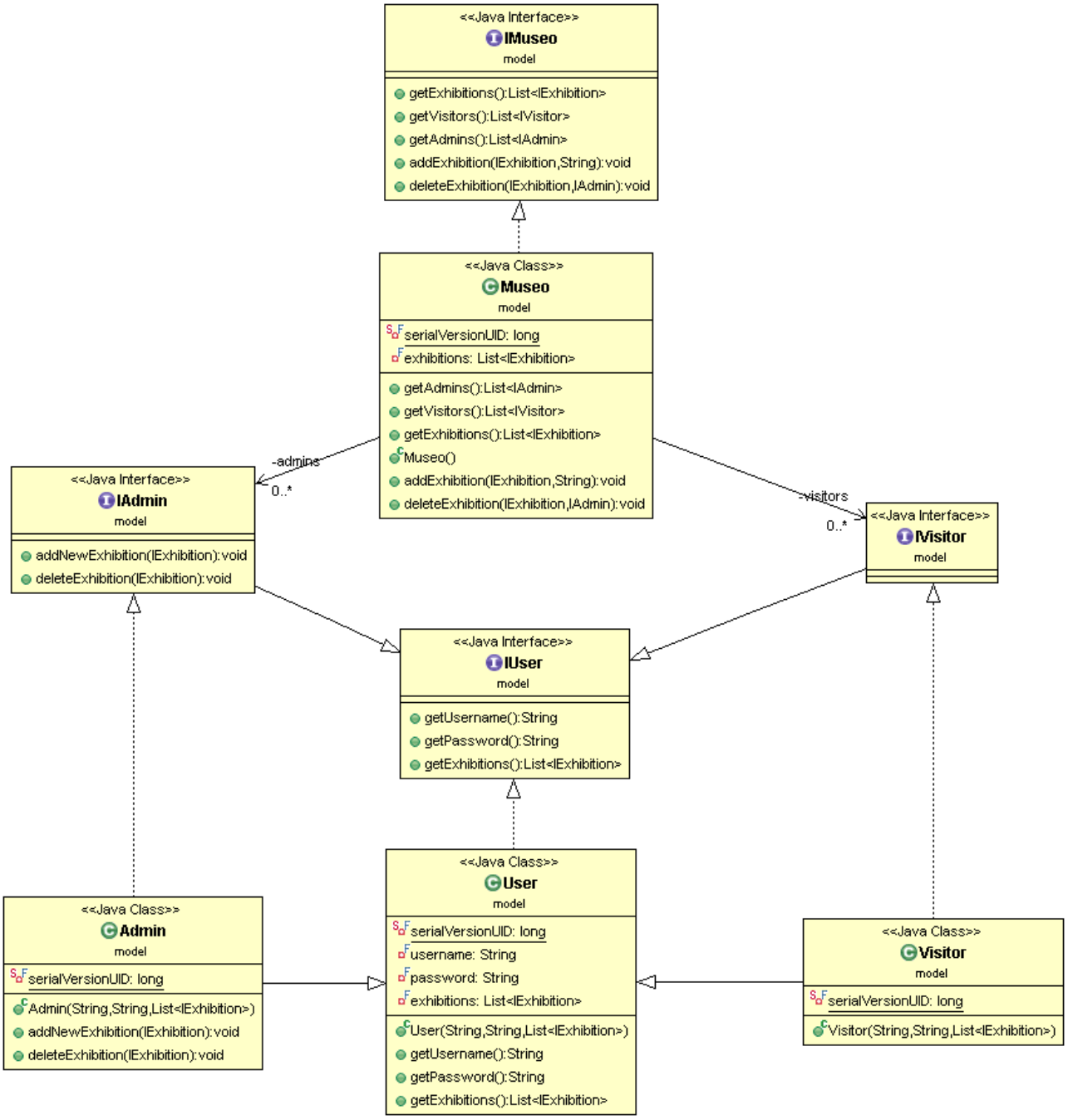
Nella fase di progettazione si è fatto un confronto tra i vari pattern architetturali, scegliendo quello che in base all'applicazione da realizzare risultava più robusto ed efficace.

L'intera progettazione è stata fondata sull'utilizzo del pattern MVC (Model – View – Controller) allo scopo di assicurare una distinzione tra modellazione e visualizzazione.

Si è così riusciti a suddividere il lavoro in due parti distinte permettendo anche, grazie al pattern usato, la possibilità in un futuro di poter estendere le funzionalità dell'applicazione.

Si descrivono attraverso diagrammi UML gli aspetti principali dell'architettura del sistema.

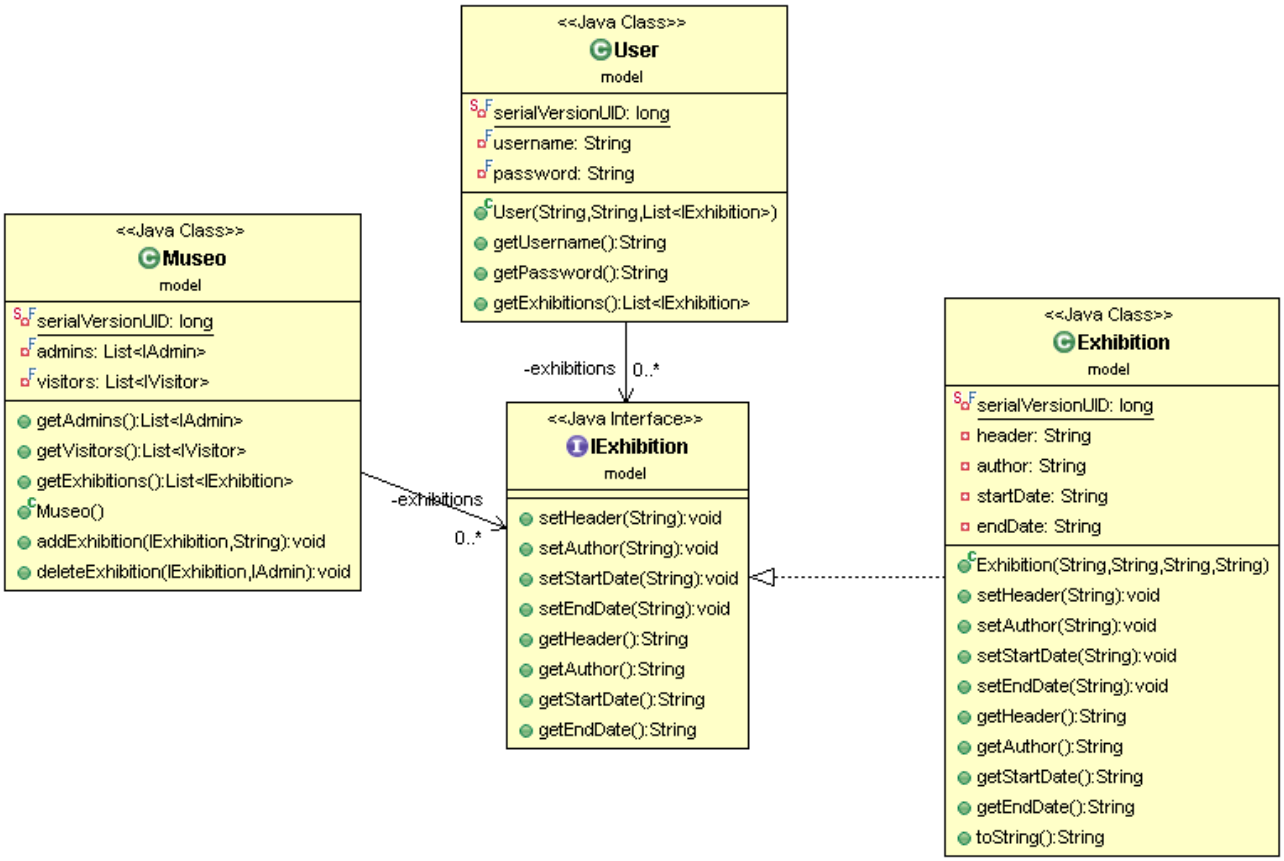
- Diagramma UML sulla parte del model, relativo all'interazione degli utenti con esso.



Descrizione degli aspetti principali:

- IMuseo e Museo: rappresentano rispettivamente l'interfaccia e l'implementazione del modello. Quest'ultima implementa l'interfaccia Serializable per garantire la persistenza dei dati.
- IUser e User: rappresentano rispettivamente l'interfaccia e l'implementazione di un generico utente (amministratore/visitatore). Quest'ultima implementa l'interfaccia Serializable per garantire la persistenza dei dati.
- IAdmin e Admin: interfaccia e implementazione di un generico amministratore. Quest'ultima implementa l'interfaccia Serializable per garantire la persistenza dei dati.
- IVisitor e Visitor: interfaccia e implementazione di un generico visitatore. Quest'ultima implementa l'interfaccia Serializable per garantire la persistenza dei dati.

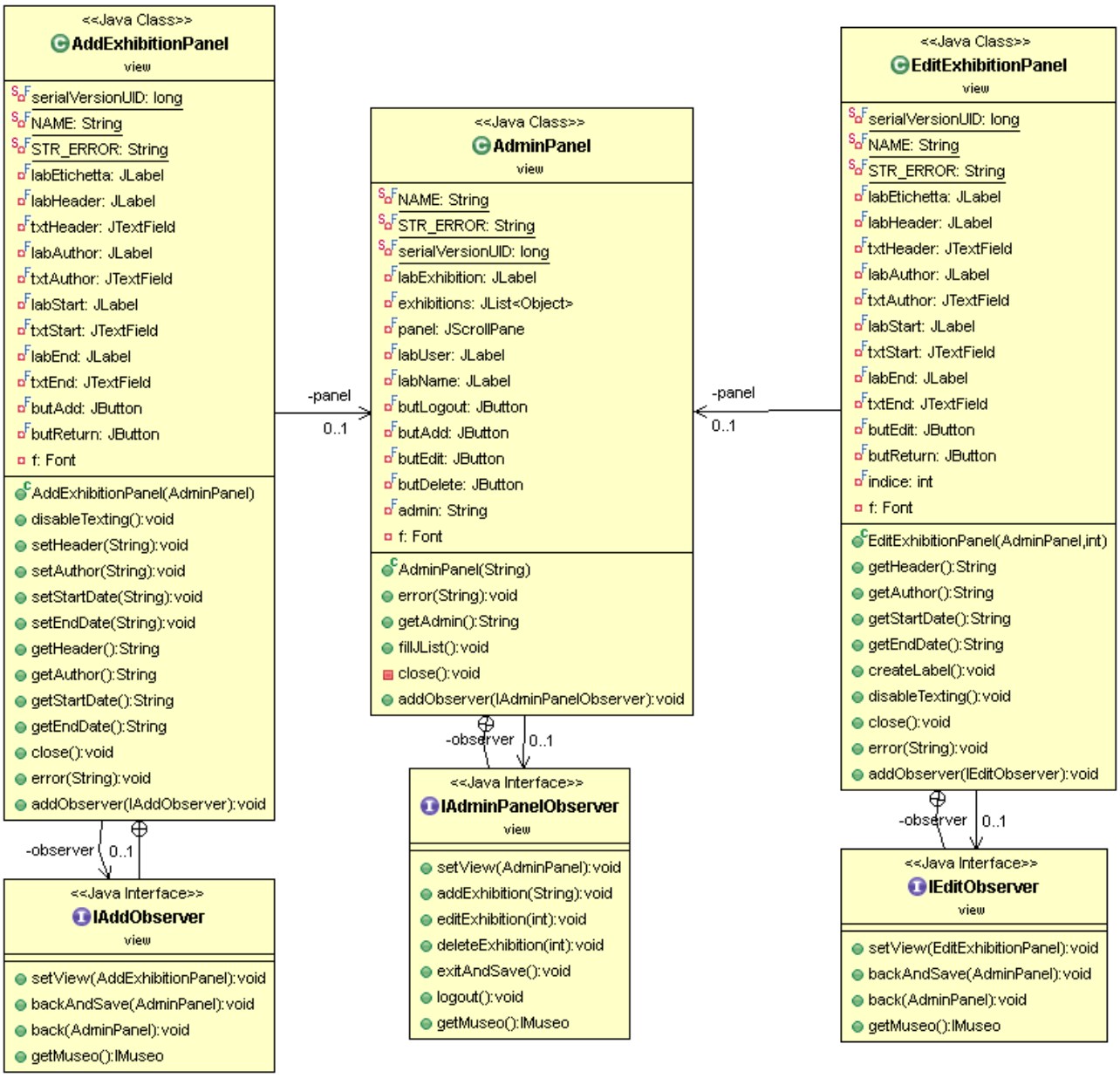
- Diagramma UML sulla parte del model, relativo all'associazione con le mostre.



Descrizione degli aspetti principali:

- IExhibition e Exhibition : rappresentano rispettivamente l'interfaccia e l'implementazione di una mostra. Quest'ultima implementa l'interfaccia Serializable per garantire la persistenza dei dati.

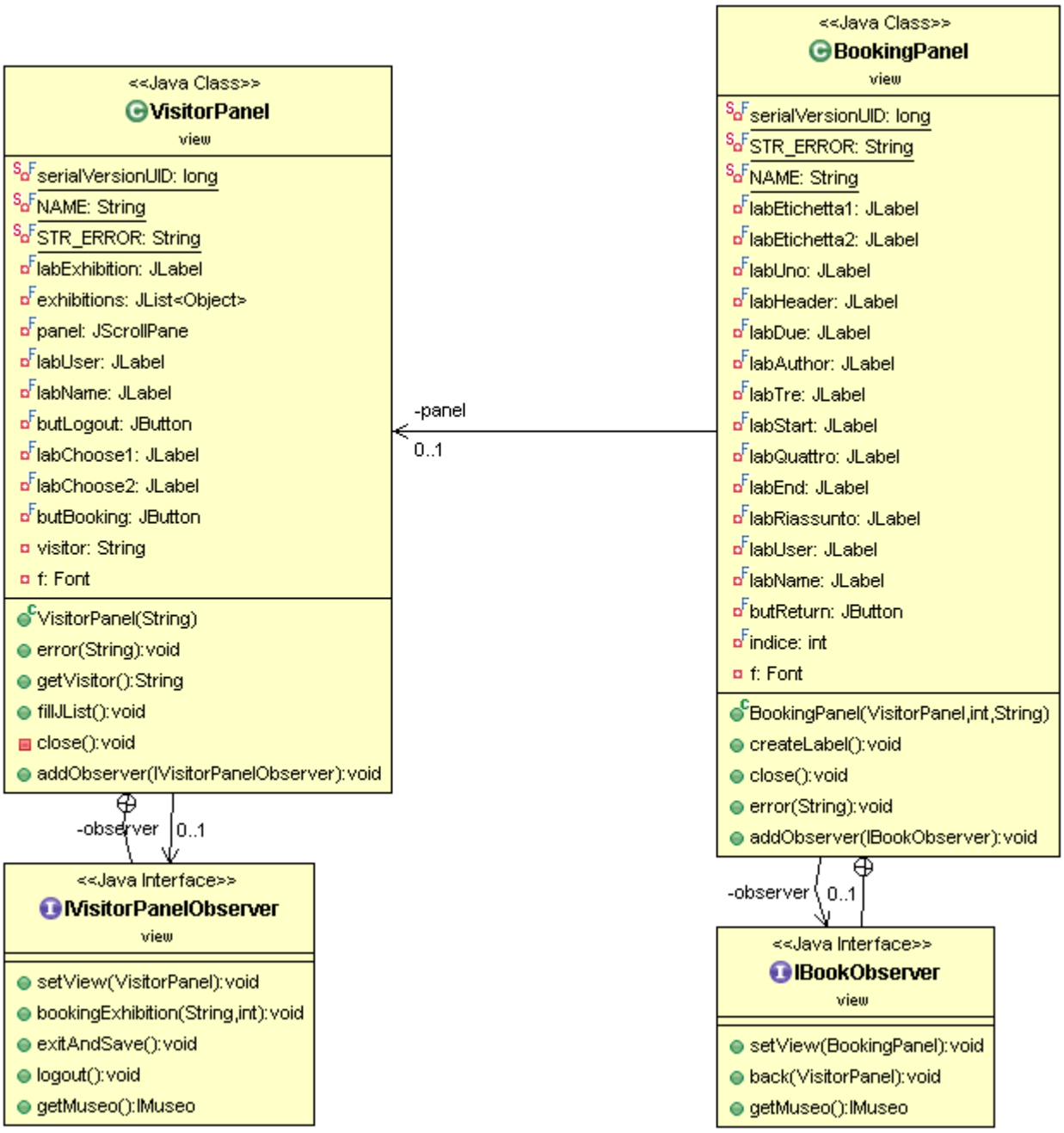
- Diagramma UML sulla parte della view, relativa all'amministratore.



Descrizione degli aspetti principali:

- AdminPanel e IAdminPanelObserver: GUI e relativa interfaccia per l'observer della vista dell'amministratore.
- AddExhibitionPanel e IAddObserver: GUI e relativa interfaccia per l'observer della vista dell'aggiunta di una nuova mostra. Questa operazione viene gestita dall'amministratore.
- EditExhibitionPanel e IEditObserver: GUI e relativa interfaccia per l'observer della vista della modifica di una mostra presente nella lista. Questa operazione viene gestita dall'amministratore.

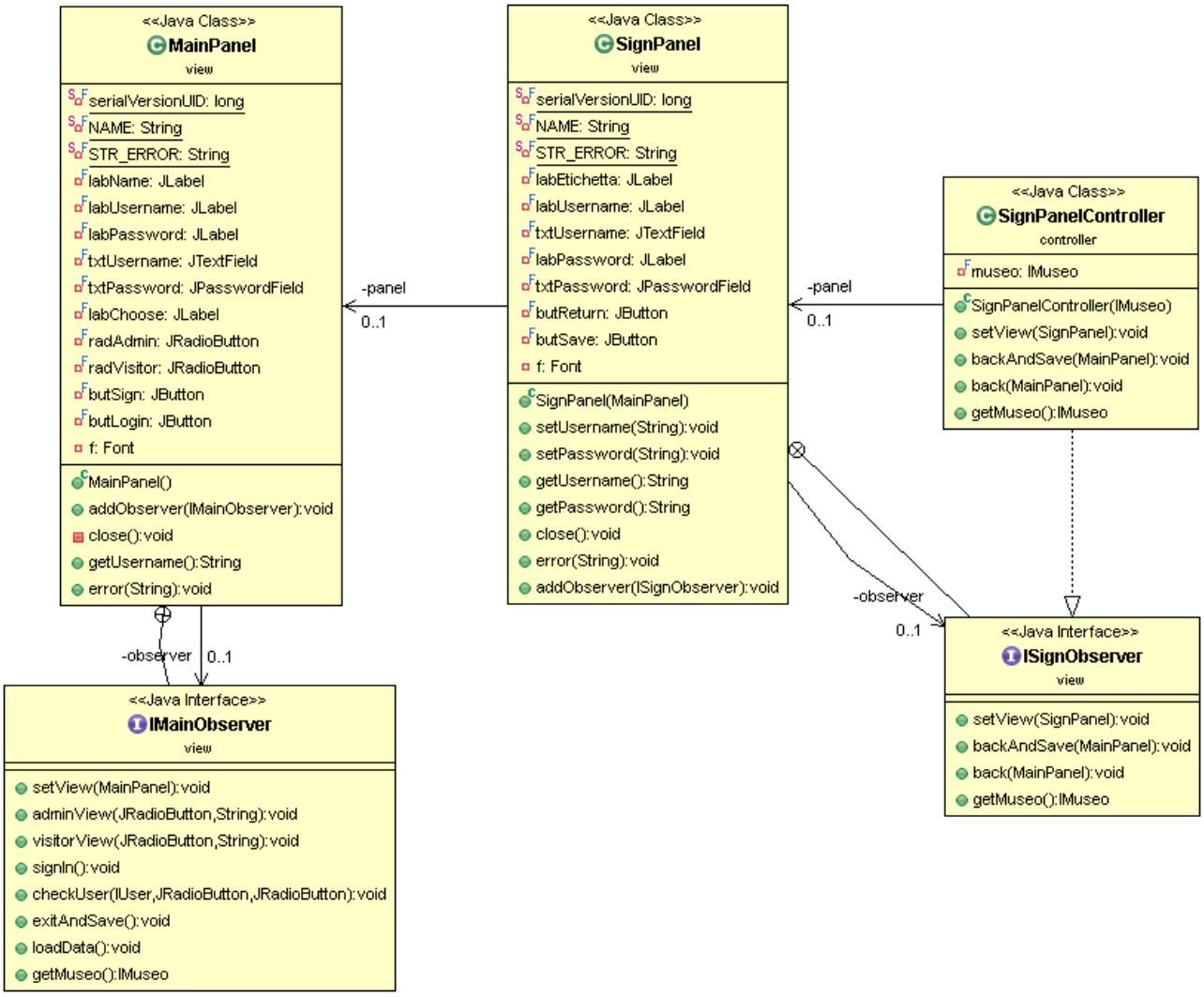
- Diagramma UML sulla parte della view, relativa al visitatore.



Descrizione degli aspetti principali:

- VisitorPanel e IVisitorPanelObserver: GUI e relativa interfaccia per l'observer della vista del visitatore.
- BookingPanel e IBookObserver: GUI e relativa interfaccia per l'observer della vista della prenotazione di una mostra del programma. Questa operazione viene gestita dal visitatore.

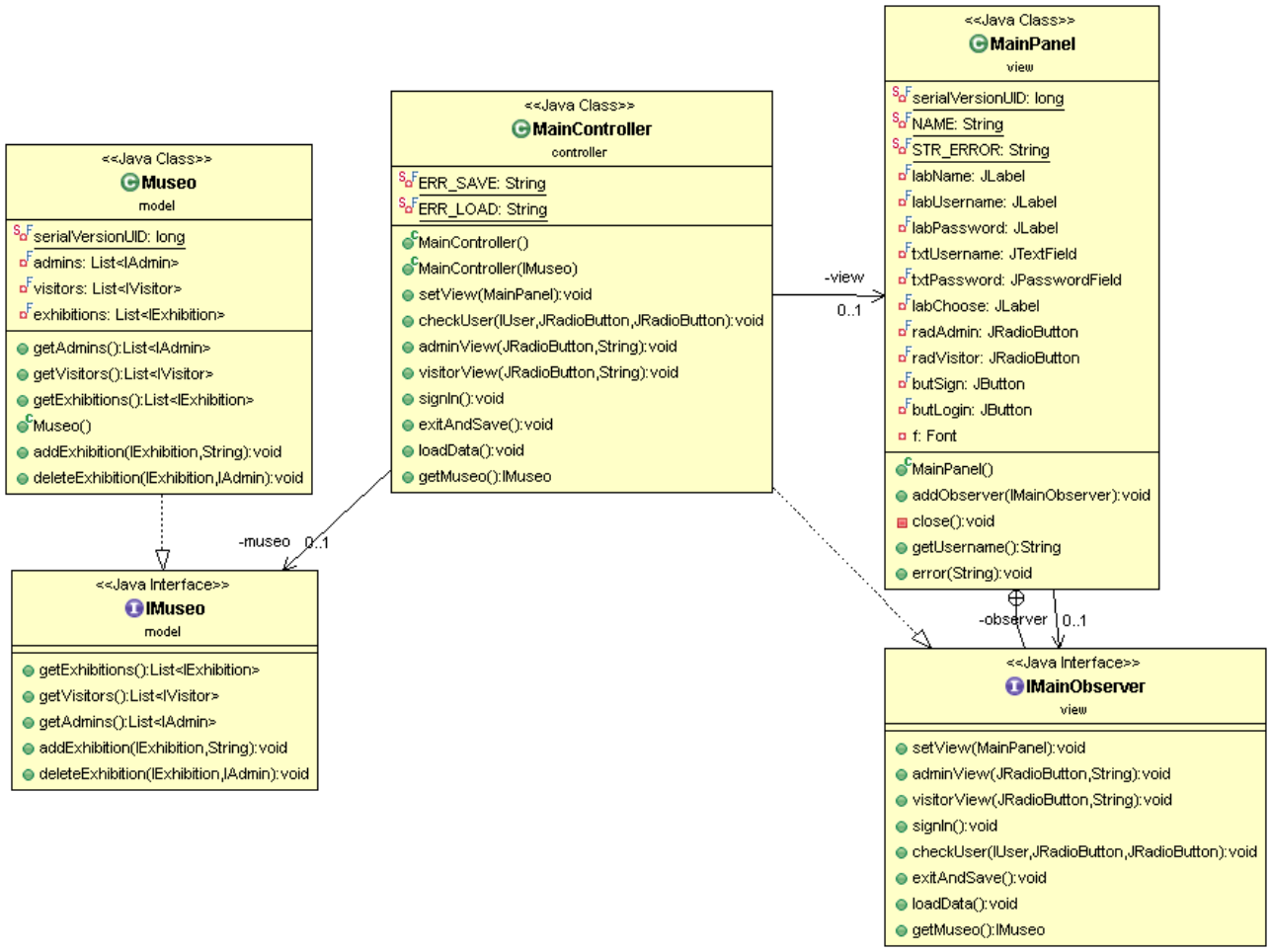
- Diagramma UML relativo alla parte di registrazione di un visitatore.



Descrizione degli aspetti principali:

- SignPanel e ISignObserver: GUI e relativa interfaccia per l'observer della vista di registrazione di un nuovo visitatore.

- Diagramma UML sull'interazione tra la parte di view, model e controller dell'applicazione, riguardo alla pagina principale 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 che provengono dalla view e modificare quindi i dati del modello.
- Questa interfaccia viene implementata da un controller principale, che si occupa di gestire i cambiamenti sia nella view che nel model, aggiornando entrambi al momento delle modifiche. In questo modo, l'unica parte ad avere un riferimento al model è il controller.

3) Organizzazione dei Package

Utilizzando il pattern MVC (Model – View – Controller) i package sono stati organizzati in modo tale da esaltarne gli aspetti, e quindi la suddivisione tra modellazione, vista e controllo.

Viene di seguito elencate l'effettiva strutturazione (riportata con i package in ordine alfabetico).

- **controller:** contiene le classi che incapsulano il comportamento dei controller, sia per quanto riguarda il MainPanel che per quanto riguarda le view degli amministratori e dei visitatori. Le classi più importanti che vi troviamo sono:
    - MainController
    - AdminPanelController
    - VisitorPanelController

- **exceptions:** al suo interno troviamo le classi relative alle eccezioni che potrebbero verificarsi durante l'uso dell'applicazione

- **main:** la classe contenuta in questo package è unica ed è la classe Main che fa partire l'applicazione

- **model:** qui si hanno le interfacce e relative classi che implementano il modello dell'applicazione. Le classi più importanti contenute in questo package sono:
    - Museo
    - Exhibition
    - User

- **view:** contiene il codice usato per implementare la GUI dell'intera applicazione. Le viste principali sono costituite dalle seguenti classi:
    - MainPanel
    - AdminPanel
    - VisitorPanel

4) Suddivisione del lavoro

Il progetto è stato portato a termine suddividendo il lavoro tra i componenti del gruppo nel seguente modo:

- Conti Laura si è occupata della parte riguardante gli amministratori, sviluppando quindi le varie funzionalità connesse (visualizzazione, aggiunta, modifica, cancellazione) sia a livello di view che di controller. Sviluppo del model relativo a amministratore, mostra. Ha realizzato anche le classi relative alle eccezioni
- Tassinari Francesca si è occupata della parte riguardante i visitatori, sviluppando quindi le varie funzionalità connesse (visualizzazione, prenotazione) sia a livello di view che di controller. Sviluppo del model relativo a visitatore, museo e utente generale. Ha sviluppato anche la view e il controller per la registrazione.

Alcune parti del progetto sono state svolte con il lavoro congiunto:

- Classe Main, view e controller della pagina inziale dell'applicazione
- Correzione finale del codice e di eventuali errori, creazione diagrammi UML.

5.1) Progettazione di dettaglio: Conti Laura

Di seguito viene descritto con maggior dettaglio la parte di progetto realizzata esclusivamente da Conti Laura.

- Descrizione della parte grafica dell'applicazione riguardante la parte degli amministratori. La GUI è stata realizzata utilizzando la classe Spring Layout.

    - **AdminPanel:** è la classe che corrisponde alla prima schermata mostrata quando il login è effettuato da un amministratore. Visualizza le mostre in programma. In alto a destra è indicato lo username dell'amministratore ed è posizionato il JButton di logout. Sono inoltre presenti altri tre JButton: il primo permette di aggiungere una nuova mostra, il secondo permette di modificare una mostra già presente, il terzo permette di cancellare una mostra dal programma.
    - **AddExhibitionPanel:** la classe è utilizzata dall'amministratore e viene richiamata quando quest'ultimo, premendo il bottone "aggiungi", vuole inserire una nuova mostra. Presenta due JButton: uno "ritorna" alla schermata dell'amministratore, l'altro inoltre memorizza i dati inseriti.
    - **EditExhibitionPanel:** la classe è utilizzata dall'amministratore e viene richiamata quando quest'ultimo, premendo il bottone "modifica", vuole modificare una mostra già presente nel programma. Presenta due JButton: uno "ritorna" alla schermata dell'amministratore, l'altro inoltre memorizza i dati modificati.
    -

- Descrizione dei controller riguardante la parte degli amministratori.

    - **AdminPanelController:** implementa l'interfaccia *IAdminPanelObserver*. Apre le diverse schermate a seconda della scelta del bottone effettuata nel pannello di amministratore. Le due schermate che si possono aprire sono quella di aggiunta o modifica, mentre la cancellazione viene gestita solo a livello di controller.
    - **AddExhibitionController:** implementa l'interfaccia *IAddObserver*. Permette di aggiungere una nuova mostra. Lancia l'eccezione nel caso in cui non tutti i campi vengano riempiti.
    - **EditExhibitionController:** implementa l'interfaccia *IEditObserver*. Permette di modificare i dati di una mostra già presente. Lancia l'eccezione nel caso in cui non tutti i campi vengano riempiti.

- Descrizione della parte di modellazione relativa a amministratore, museo.

    - **Admin:** la classe rappresenta la modellazione dell'utente di tipo amministratore. Presenta le sue funzionalità quali aggiungere e cancellare mostre. La modifica viene gestita a livello di view. Estende la classe User e implementa l'interfaccia *IAdmin* e *Serializable*. Quest'ultima assicura la persistenza dei dati.
    - **Exhibition:** è la classe di base che descrive la struttura di una mostra. Presenta un titolo, l'autore, la data di inizio e la data di fine. Ha metodi setter e getter per impostare e recuperare i dati della mostra. Implementa il metodo toString, che viene usato per stampare

le informazioni riguardanti la mostra all'interno della JList presente in due view. Implementa l'interfaccia *IExhibition* e *Serializable*. Quest'ultima assicura la persistenza dei dati.

- Descrizione della parte riguardante le eccezioni.

  - **EmptyFieldExc:** la classe rappresenta l'eccezione che viene lanciata quando, durante l'inserimento di una nuova mostra o durante la registrazione di un utente, non vengono riempiti tutti i campi.
  - **ExhibitionNotSelectedExc:** la classe rappresenta l'eccezione che viene lanciata quando l'amministratore non seleziona una mostra prima di premere il bottone "modifica" e "cancella", o quando il visitatore non seleziona una mostra prima di premere il bottone "prenota".
  - **LoginExc:** la classe rappresenta l'eccezione che viene lanciata quando esiste la combinazione username e password, ma non è stato selezionato il giusto tipo di utente corrispondente.
  - **UserNotFoundExc:** la classe rappresenta l'eccezione che viene lanciata quando username e password non coincidono con nessuna combinazione esistente nelle liste di amministratori e visitatori.

5.2) Progettazione di dettaglio: Tassinari Francesca

Di seguito viene descritto con maggior dettaglio la parte di progetto realizzata esclusivamente da Tassinari Francesca.

- Descrizione della parte grafica dell'applicazione riguardante la parte dei visitatori. La GUI è stata realizzata utilizzando la classe Spring Layout.

  - **VisitorPanel:** è la classe che corrisponde alla prima schermata mostrata quando il login è effettuato da un visitatore. Visualizza le mostre in programma. In alto a destra è indicato lo username del visitatore ed è posizionato il JButton di logout. È presente un altro JButton che permette di prenotare una mostra.
  - **BookingPanel:** la classe è utilizzata dal visitatore e viene richiamata quando quest'ultimo, premendo il bottone "prenota", vuole prenotare una mostra del programma. Questa schermata visualizza il riassunto dei dati riguardanti la mostra prenotata e indica lo username del visitatore che ha effettuato la prenotazione. Presenta un JButton per ritornare alla schermata principale del visitatore. Non sono presenti JButton, in quanto il visitatore deve a questo punto stampare la schermata e presentare il foglio stampato alla cassa del museo per ottenere uno sconto del 20%.
  - **SignPanel:** la classe viene utilizzata dal visitatore che, prima di poter accedere all'applicazione, deve registrarsi. La schermata mostra due semplici campi di inserimento, username e password, e due JButton: uno "ritorna" alla schermata iniziale, l'altro permette inoltre di memorizzare i dati inseriti.

- Descrizione dei controller riguardante la parte dei visitatori.

    - **VisitorPanelController:** implementa l'interfaccia *IVisitorPanelObserver*. Una volta che il visitatore seleziona il film da prenotare viene aperta una schermata relativa alla prenotazione.
    - **BookingPanelController:** implementa l'interfaccia *IBookObserver*. Non permette l'apertura di nuove schermate, ma soltanto il ritorno a quella precedente.
    - **SignPanelController:** implementa l'interfaccia *ISignObserver*. Permette di registrare un nuovo visitatore, ma non apre nuove schermate. Lancia l'eccezione nel caso in cui non tutti i campi vengano riempiti.

- Descrizione della parte di modellazione relativa a visitatore, mostra, utente generale.

    - **Visitor:** la classe rappresenta la modellazione dell'utente di tipo visitatore. Estende la classe User e implementa l'interfaccia *IVisitor* e *Serializable*. Quest'ultima assicura la persistenza dei dati.
    - **Museo:** rappresenta il vero e proprio modello. Possiede due liste dove vengono memorizzati gli amministratori e i visitatori che usano l'applicazione. Questi vengono istanziati nel costruttore della classe e memorizzati nelle rispettive liste. Una terza lista memorizza le mostre programmate. Offre due metodi per aggiungere e cancellare le mostre che sono utilizzate dagli amministratori. Implementa l'interfaccia *IMuseo* e *Serializable*. Quest'ultima assicura la persistenza dei dati.
    - **User:** la classe rappresenta la modellazione di un generico utente che usa l'applicazione. Presenta pertanto tre campi costituiti da username, password e la lista delle mostre. Questa classe viene estesa per modellare dettagliatamente due diversi tipi di utenti, che sono l'amministratore e il visitatore. Implementa l'interfaccia *IUser* e *Serializable*. Quest'ultima assicura la persistenza dei dati.

5.3) Progettazione di dettaglio: Conti Laura e Tassinari Francesca

In comune è stata realizzata la classe Main, la schermata principale dell'applicazione (MainPanel), e la classe MainController che gestisce il login degli utenti verificando la validità di username, password e scelta del tipo di utente.
Al termine della realizzazione del progetto è stato svolta insieme una verifica del suo funzionamento e la correzione di eventuali errori.
È stata inoltre svolta in comune la realizzazione dei diagrammi UML.

6) Note

- **Modalità di lavoro**

Nella proposta di progetto la suddivisione dei compiti era differente. In fase di progettazione, a seguito di una maggiore concezione del lavoro complessivo da svolgere, i compiti sono stati redistribuiti nel modo indicato in precedenza. È stata preferita una divisione in modo tale da poter entrambe sviluppare una parte del model, della view e del controller. La parte delle eccezioni e della registrazione sono state divise in modo da poter avere un lavoro equamente distribuito. L'ammontare delle ore svolte in comune rimane inferiore al 30%.

- **Note sull'applicazione**

Le credenziali di login per quanto riguarda gli amministratori sono già dall'inizio state inserite in una lista. Tale lista non è modificabile e viene istanziata al momento della creazione del museo, nel suo costruttore. Di seguito vengono indicate le credenziali degli amministratori per poter accedere all'applicazione.

Credenziali degli amministratori:

Username: Laura
Password: Conti

Username: Francesca
Password: Tassinari

Le credenziali di login per quando riguarda i visitatori sono inserite all'interno di una lista che viene istanziata al momento della creazione del museo nel suo costruttore. In questo caso ci è sembrato comodo inserire già un visitatore, lasciando però la possibilità di modificare la lista con l'aggiunta di nuovi utenti che si potranno registrare.

Credenziali del visitatore:

Username: user1
Password: pwd1