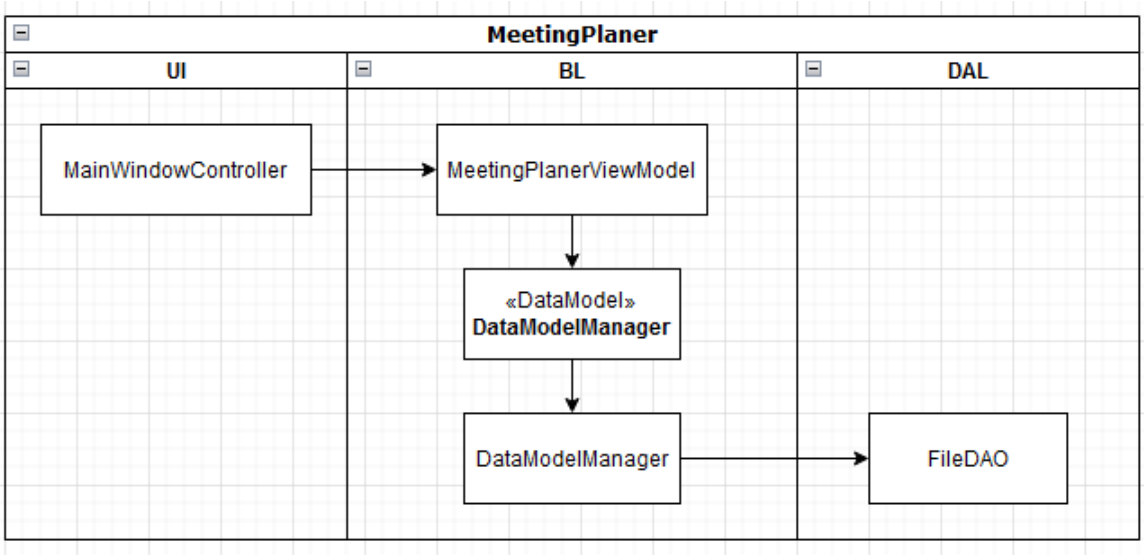
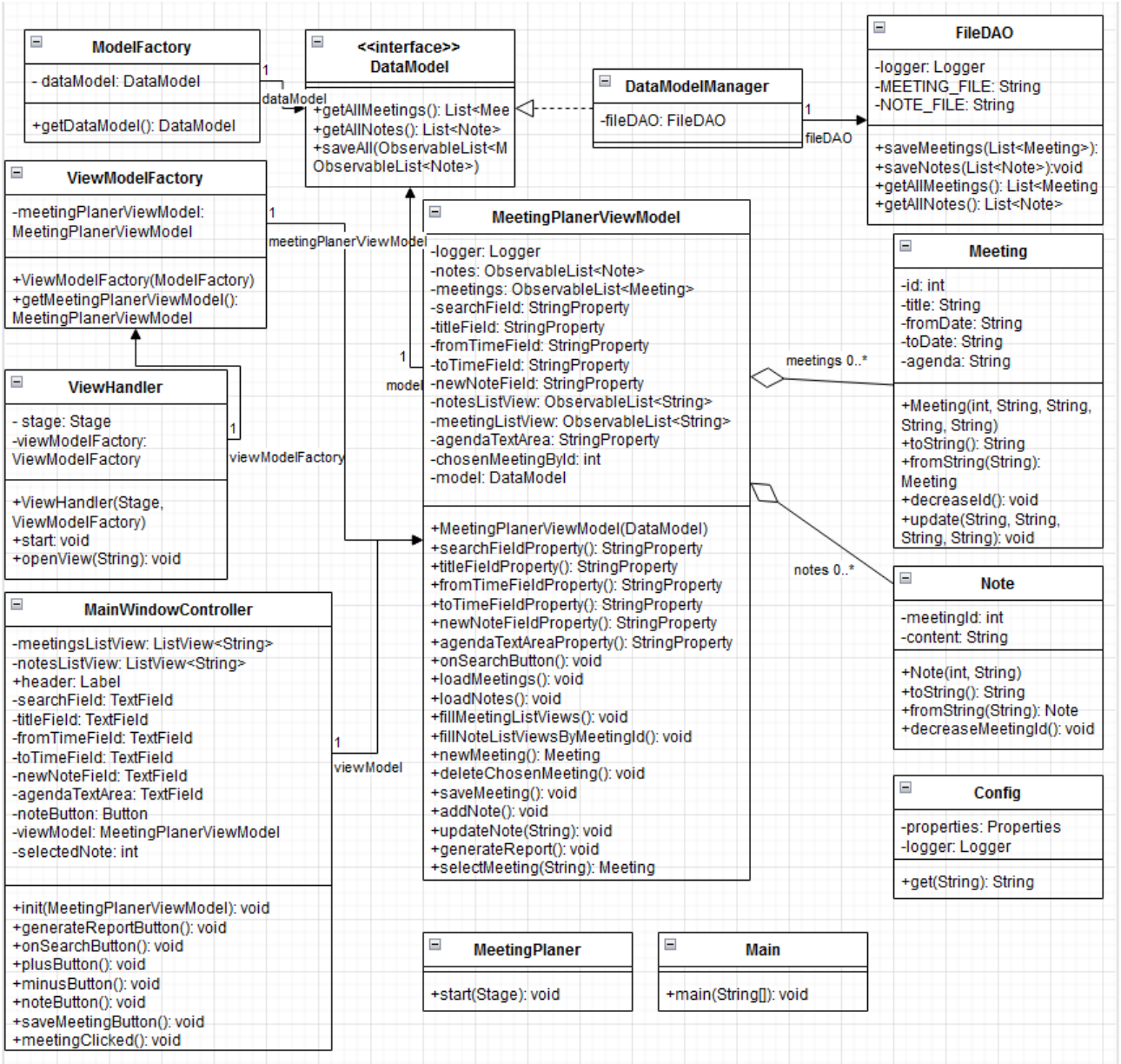

MeetingPlaner
UI
BL
DAL
MainWindowController
MeetingPlanerViewModel
«DataModel»
DataModelManager
DataModelManager
FileDAO


ModelFactory
- dataModel: DataModel
+getDataModel(): DataModel
1
dataModel
<<interface>>
DataModel
+getAllMeetings(): List<Mee
+getAllNotes(): List<Note>
+saveAll(ObservableList<M
ObservableList<Note>)
DataModelManager
-fileDAO: FileDAO
1
fileDAO
FileDAO
-logger: Logger
-MEETING_FILE: String
-NOTE_FILE: String
+saveMeetings(List<Meeting>):
+saveNotes(List<Note>):void
+getAllMeetings(): List<Meeting
+getAllNotes(): List<Note>
ViewModelFactory
-meetingPlanerViewModel:
MeetingPlanerViewModel
+ViewModelFactory(ModelFactory)
+getMeetingPlanerViewModel():
MeetingPlanerViewModel
1
meetingPlanerViewModel
MeetingPlanerViewModel
-logger: Logger
-notes: ObservableList<Note>
-meetings: ObservableList<Meeting>
-searchField: StringProperty
-titleField: StringProperty
-fromTimeField: StringProperty
-toTimeField: StringProperty
-newNoteField: StringProperty
-notesListView: ObservableList<String>
-meetingListView: ObservableList<String>
-agendaTextArea: StringProperty
-chosenMeetingById: int
-model: DataModel
+MeetingPlanerViewModel(DataModel)
+searchFieldProperty(): StringProperty
+titleFieldProperty(): StringProperty
+fromTimeFieldProperty(): StringProperty
+toTimeFieldProperty(): StringProperty
+newNoteFieldProperty(): StringProperty
+agendaTextAreaProperty(): StringProperty
+onSearchButton(): void
+loadMeetings(): void
+loadNotes(): void
+fillMeetingListViews(): void
+fillNoteListViewsByMeetingId(): void
+newMeeting(): Meeting
+deleteChosenMeeting(): void
+saveMeeting(): void
+addNote(): void
+updateNote(String): void
+generateReport(): void
+selectMeeting(String): Meeting
1
model
meetings 0..*
Meeting
-id: int
-title: String
-fromDate: String
-toDate: String
-agenda: String
+Meeting(int, String, String,
String, String)
+toString(): String
+fromString(String):
Meeting
+decreaseId(): void
+update(String, String,
String, String): void
notes 0..*
Note
-meetingId: int
-content: String
+Note(int, String)
+toString(): String
+fromString(String): Note
+decreaseMeetingId(): void
ViewHandler
- stage: Stage
-viewModelFactory:
ViewModelFactory
+ViewHandler(Stage,
ViewModelFactory)
+start: void
+openView(String): void
1
viewModelFactory
MainWindowController
-meetingsListView: ListView<String>
-notesListView: ListView<String>
+header: Label
-searchField: TextField
-titleField: TextField
-fromTimeField: TextField
-toTimeField: TextField
-newNoteField: TextField
-agendaTextArea: TextField
-noteButton: Button
-viewModel: MeetingPlanerViewModel
-selectedNote: int
+init(MeetingPlanerViewModel): void
+generateReportButton(): void
+onSearchButton(): void
+plusButton(): void
+minusButton(): void
+noteButton(): void
+saveMeetingButton(): void
+meetingClicked(): void
1
viewModel
Config
-properties: Properties
-logger: Logger
+get(String): String
MeetingPlaner
+start(Stage): void
Main
+main(String[]): void

Design patterns:
For the design patterns I used

- MVVM design pattern, it bridges the View and the Model. The Model Represents the Data. The View interacts with the user via JavaFX components. The ViewModel manages the data bindings and application logic.
- Observer design patter, it triggers a function when something specific happens. I added a Listener for the notesListView so when one Note is clicked/selected then the ID will be saved and the Button text will change
- Singleton design pattern, it makes sure that only one Instance can exist of it. To improve performance, memory usage, and consistency. I used it for the FileDAO
- Factory design pattern, it centralizes object creation logic to ensure consistent initialization and encapsulates object creation. I used it for ViewHandler to create MainWindowController, ViewModelFactory to create MeetingPlanerViewModel, and for ModelFactory to create DataModelManager.

Unit Tests:
onSearchButton_*: ensures seamless communication between the data model and the UI
fillMeetingListViews_*: ensures the UI reflectis the underlying meeting data accurately
newMeeting: validate that core functionalities(e.g. CRUD) behave as intended
deleteChosenMeeting: same as above
saveMeeting_withEmptyString: Ensures stability with incomplete user input
saveMeeting_samTitle: Prevents duplicates, critical for data integrity
updateNote_*: validate note modification behaviors

Git:
https://github.com/if23b118/MeetingPlanner.git