

Tesina di Ingegneria del Software

Università degli studi di Modena e Reggio Emilia

Filippo Leonelli
MATR. n° 164823

Dichiaro che questo elaborato è frutto del mio personale lavoro, svolto
sostanzialmente in maniera individuale e autonoma

Sommario:

1 - Introduzione:	3
1.1 - Premesse e Obiettivo del documento:	3
2 - Descrizione generale:	4
2.1 - Interfaccia hardware:	4
2.2 - Interfaccia software e utente:	4
2.3 - Vincoli:	5
2.4 - Caratteristiche e funzionalità principali:	5
2.5 - Possibili aggiornamenti futuri:	5
3 - Specifica dei requisiti:	6
3.1 - Requisiti funzionali:	6
3.1.1 - Caricamento dei file video/audio locali:	6
3.1.2 - Riproduzione contenuto:	6
3.1.3 - Pausa contenuto:	7
3.1.4 - Interruzione contenuto:	7
3.1.5 - Modifica alla velocità di riproduzione del contenuto:	8
3.1.6 - Modifica del volume del contenuto:	8
3.1.7 - Modifica della riproduzione del contenuto:	9
3.1.8 - Zoom in dell'applicazione:	9
3.1.9 - Zoom out dell'applicazione:	10
3.2 - Requisiti non funzionali:	10
3.2.1 - Facilità di utilizzo:	10
3.2.2 - Affidabilità:	11
3.2.3 - Performance:	11
4 - Design e struttura:	12
4.1 - Use case diagram:	12
4.2 - Class Diagram:	13
4.2.1 - Descrizione class diagram:	14
4.3 - Sequence diagram:	15
4.4 - Activity diagram:	16
4.5 - State diagram:	17
5 - Design pattern:	18
5.1 - Abstract Factory di MediaPlayer:	18
5.2 - Observer per ObservableMediaPlayer:	19
5.3 - Strategy per le estensioni dei file:	20

1 - Introduzione:

Il seguente documento si pone l'obiettivo di presentare, con una visione completa dei principali aspetti, il software FH_MediaPlayer, un semplice tool per il consumo di contenuti video e audio. Il seguente documento rappresenta la Specifica dei Requisiti (SRS) del progetto secondo lo standard ANSI/IEEE 830.

1.1 - Premesse e Obiettivo del documento:

L'obiettivo di questo documento è quello di presentare nella maniera più completa possibile la progettazione e la struttura di FH_MediaPlayer.

FH_MediaPlayer è nato con l'intenzione di essere un puro progetto per l'apprendimento di alcune categorie di oggetti specifici di JavaFx e per poter creare un tool personale per la fruizione di contenuti video e audio.

Come si vedrà anche in seguito, verranno esposte tecniche tipiche dell'ingegneria del software per rendere tale prodotto mantenibile ed evolvibile nel tempo.

2 - Descrizione generale:

FH_MediaPlayer è una semplice applicazione di riproduzione di file video e audio che permette all'utente di poter riprodurre i propri contenuti preferiti con semplicità. Nonostante sia un progetto molto semplice e nato con il puro scopo di apprendere nozioni in più su JavaFx, rimane molto aperto a delle estensioni e aggiornamenti futuri (quali miglioramenti della user experience e introduzione di altri formati per i file) .

2.1 - Interfaccia hardware:

Il software è stato sviluppato per essere il più leggero possibile e in generale può essere eseguito su qualsiasi dispositivo possa supportare il Java Runtime Environment (JRE). Richiede risorse hardware veramente esigue e non ha bisogno di una connessione ad internet per poter funzionare in maniera corretta.

Just download and play!

2.2 - Interfaccia software e utente:

FH_MediaPlayer è stato concepito per interfacciarsi dal punto di vista software solamente con il sistema operativo su cui sta operando, per permettere all'utente di poter selezionare il proprio file da riprodurre. Quest'ultima caratteristica verrà esplicitata in seguito. L'intero software è stato scritto in Java 17 con utilizzo per l'interfaccia grafica di JavaFx.

Per quanto riguarda la user experience, all'utente non sono richieste particolari conoscenze informatiche per poter fruire al meglio dei servizi offerti dal software in questione. L'interfaccia utente risulta particolarmente semplice ed intuitiva proprio per rendere facile l'utilizzo del software ad ogni tipo di utente che vi si interfacci.

Simple, enjoy!

2.3 - Vincoli:

FH_MediaPlayer non ha particolari vincoli di esecuzione se non quelli espressi in precedenza per quanto riguarda il JRE. Per eseguire tale software senza incorrere in problemi, è necessario infatti avere un dispositivo desktop o laptop (non mobile) che possieda una versione del JRE pari o superiore alla 17. Inoltre non è consigliabile eseguire tale applicazione su sistemi embedded dato che non vi è alcuna garanzia di funzionamento.

2.4 - Caratteristiche e funzionalità principali:

Nel sistema è presente un solo tipo di utente che, una volta avviata l'applicazione, è in grado di poter eseguire ogni funzionalità presente nel sistema stesso. L'utente che sta eseguendo il software, in ogni caso, deve avere la possibilità di accesso al file manager del proprio sistema operativo.

Le principali funzionalità del software sono:

- 1) Riprodurre file video MP4.
- 2) Riprodurre file audio MP3.
- 3) Avviare, mettere in pausa e riprodurre dall'inizio file video/audio.
- 4) Velocizzare o rallentare su più scale la velocità del contenuto.
- 5) Regolare il volume del contenuto.
- 6) Modificare la riproduzione del file.

2.5 - Possibili aggiornamenti futuri:

Per il futuro si potrebbe migliorare la parte grafica e di user experience dell'utente, rendendo ancora più piacevole alla vista l'intera applicazione e migliorarne l'utilizzo. Inoltre si potrebbero aggiungere altri formati video e audio, nonché formati per file immagini in modo da rendere ancora più completa e general purpose (per quanto riguarda la fruizione di contenuti) l'applicazione. Quest'ultimo tipo di aggiornamento sarebbe facilitato dall'utilizzo fatto in fase di progettazione e poi di realizzazione, di design patterns (vedi sezione 5).

3 - Specifica dei requisiti:

3.1 - Requisiti funzionali:

3.1.1 - Caricamento dei file video/audio locali:

RF01	Caricamento dei file video/audio locali
Input	L'utente clicca sul pulsante "Open File".
Processo	Il sistema apre il file manager del relativo sistema operativo su cui è eseguito, e permette all'utente di selezionare un file da caricare nell'applicazione (al momento solo MP4 e MP3).
Output	Se l'utente seleziona correttamente un file e il file non è corrotto, allora il sistema lo apre e lo comincia già a riprodurre.

3.1.2 - Riproduzione contenuto:

RF02	Riproduzione contenuto
Input	L'utente clicca sul pulsante "Play".
Processo	Il sistema mette in riproduzione il contenuto video/audio e mantiene l'aggiornamento della barra di progresso del contenuto stesso.
Output	La riproduzione del contenuto ha inizio. Se era stata precedentemente messa in pausa o interrotta, riprende.

3.1.3 - Pausa contenuto:

RF03	Pausa contenuto
Input	L'utente clicca sul pulsante "Pause".
Processo	Il sistema mette in pausa la riproduzione del contenuto video/audio e mantiene l'aggiornamento della barra di progresso del contenuto stesso.
Output	La riproduzione del contenuto viene messa in pausa.

3.1.4 - Interruzione contenuto:

RF04	Interruzione contenuto
Input	L'utente clicca sul pulsante "Stop".
Processo	Il sistema ferma la riproduzione del contenuto e riporta il video/audio all'inizio. Aggiorna di conseguenza la barra di stato del contenuto.
Output	La riproduzione del contenuto viene interrotta e si riporta all'inizio il video/audio.

3.1.5 - Modifica alla velocità di riproduzione del contenuto:

RF05	Modifica alla velocità del contenuto
Input	L'utente clicca sul pulsante ">>" o ">>>" o "<<" o "<<<".
Processo	<p>Il sistema a seconda del pulsante premuto velocizza/rallenta la riproduzione del contenuto con un determinato tasso.</p> <p>Tassi di modifica:</p> <ul style="list-style-type: none">- ">>" : velocizzazione di un fattore 1.5- ">>>" : velocizzazione di un fattore 2- "<<" : rallentamento di un fattore 0.75- "<<<" : rallentamento di un fattore 0.5
Output	La riproduzione del contenuto subisce una modifica alla propria velocità (aumenta/diminuisce).

3.1.6 - Modifica del volume del contenuto:

RF06	Modifica del volume del contenuto
Input	L'utente sposta il cursore dello slider del volume.
Processo	Il sistema in base al posizionamento del cursore, modifica il volume del contenuto in riproduzione.
Output	Il contenuto in riproduzione subisce una variazione di volume.

3.1.7 - Modifica della riproduzione del contenuto:

RF07	Modifica della riproduzione del contenuto
Input	L'utente sposta il cursore dello slider della barra di progresso del contenuto.
Processo	Il sistema in base al posizionamento del cursore, sposta la riproduzione del contenuto in quel punto.
Output	La riproduzione del contenuto video/audio proseguirà dal punto indicato dall'utente attraverso il cursore

3.1.8 - Zoom in dell'applicazione:

RF08	Zoom in dell'applicazione
Input	L'utente clicca sul pulsante "Zoom in" o clicca due volte sopra lo schermo.
Processo	Il sistema, se si trova in modalità schermo ridotto (quella di default), ingrandisce l'applicazione a schermo intero. Inoltre si effettua un cambio del testo del pulsante in "Zoom out".
Output	L'applicazione passa a schermo intero.

3.1.9 - Zoom out dell'applicazione:

RF09	Zoom out dell'applicazione
Input	L'utente clicca sul pulsante "Zoom out" o preme il pulsante "Esc" della tastiera.
Processo	Il sistema, se si trova in modalità schermo intero, riduce l'applicazione a schermo ridotto. Inoltre si effettua un cambio del testo del pulsante in "Zoom in".
Output	L'applicazione passa a schermo ridotto.

3.2 - Requisiti non funzionali:

3.2.1 - Facilità di utilizzo:

RNF01	Facilità di utilizzo
Descrizione	L'applicazione deve essere semplice da utilizzare per ogni tipo di utente, anche quelli con poche competenze informatiche.

3.2.2 - Affidabilità:

RNF02	Affidabilità
Descrizione	L'applicazione deve essere affidabile durante l'intero ciclo di utilizzo da parte dell'utente. Non deve corrompere i file che vengono caricati sopra di essa, non deve subire crash casuali o in generale rendere non gradevole la user experience.

3.2.3 - Performance:

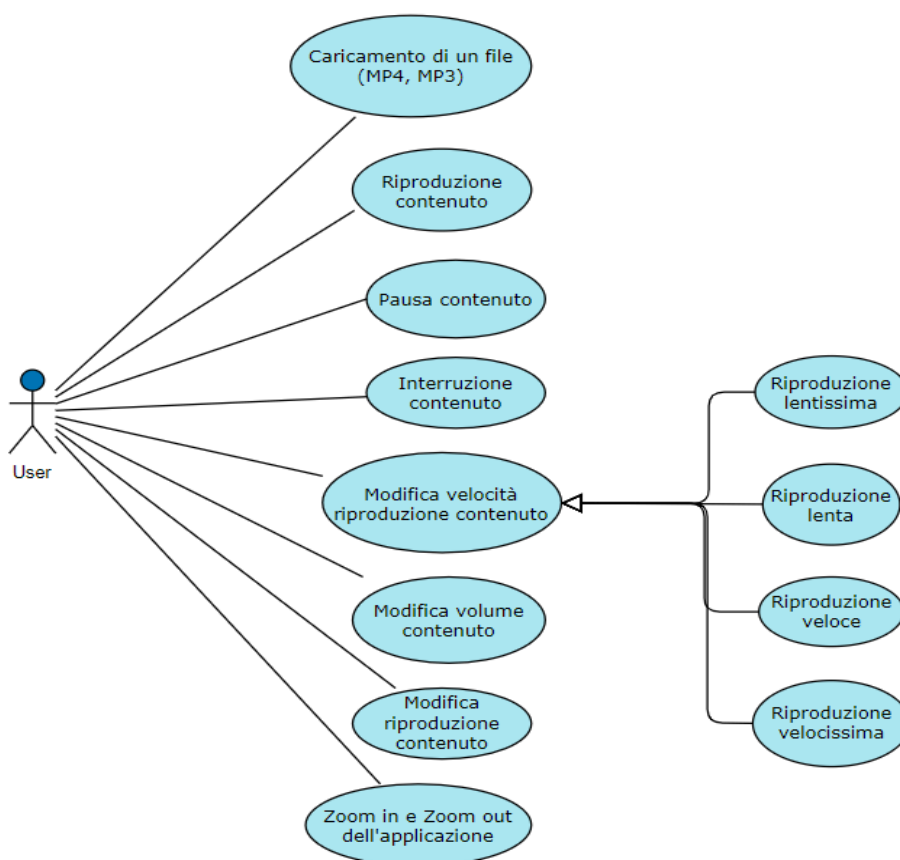
RNF03	Performance
Descrizione	L'applicazione deve garantire un certo livello di performance per poter riprodurre al meglio i contenuti richiesti dall'utente. Le basse risorse hardware che questo software richiede, permettono la riproduzione di file video/audio anche molto pesanti.

4 - Design e struttura:

In questa sezione si andrà a mostrare l'intera struttura progettuale e di design dell'applicazione. Si farà uso di diagrammi UML per fornire chiare informazioni sul progetto.

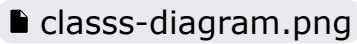
4.1 - Use case diagram:

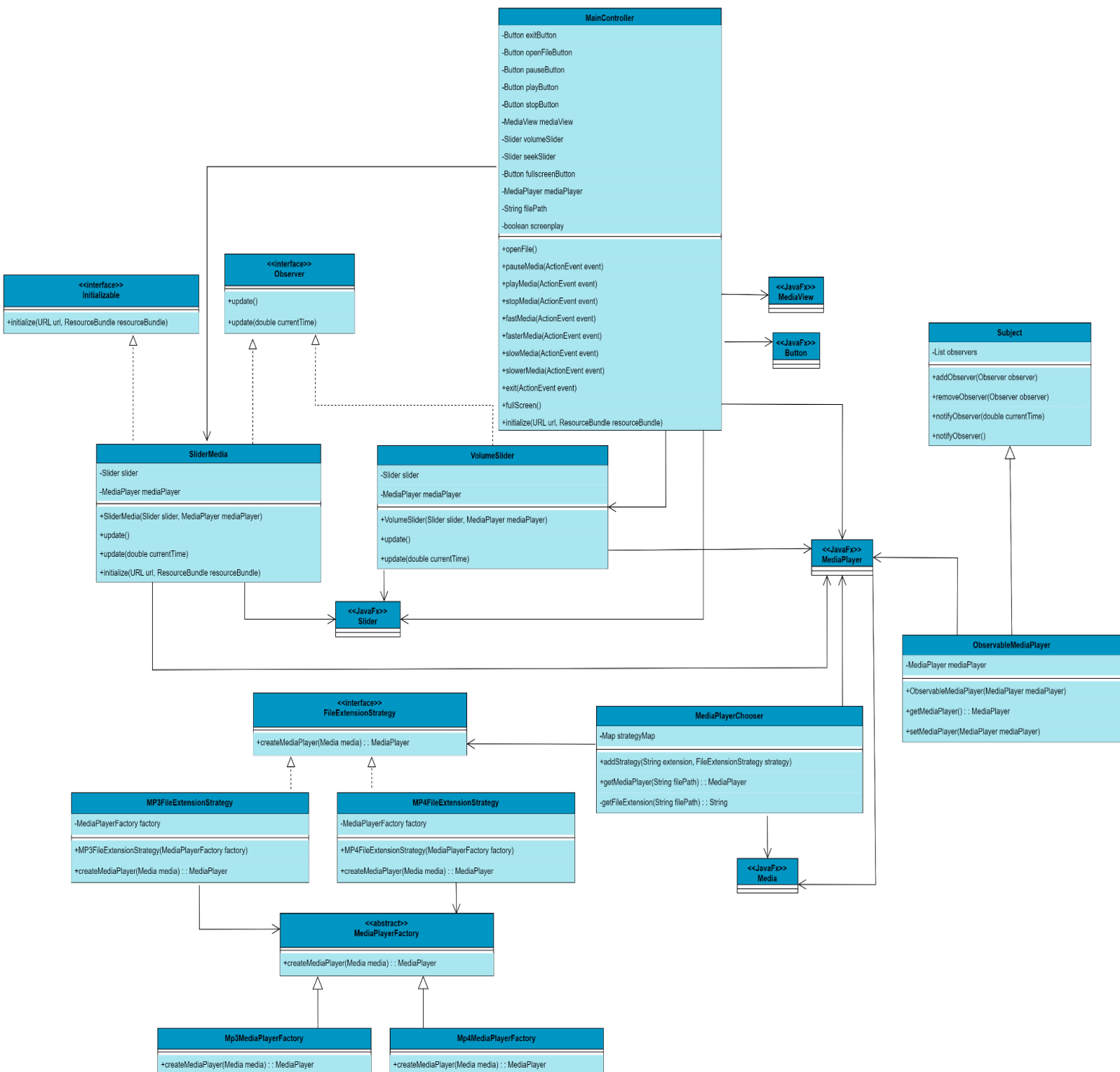
Di seguito viene mostrato lo use cas diagram dell'applicazione:



4.2 - Class Diagram:

Di seguito viene mostrato il class diagram dell'applicazione. Per poter visualizzare meglio il contenuto, è consigliabile seguire il seguente link:





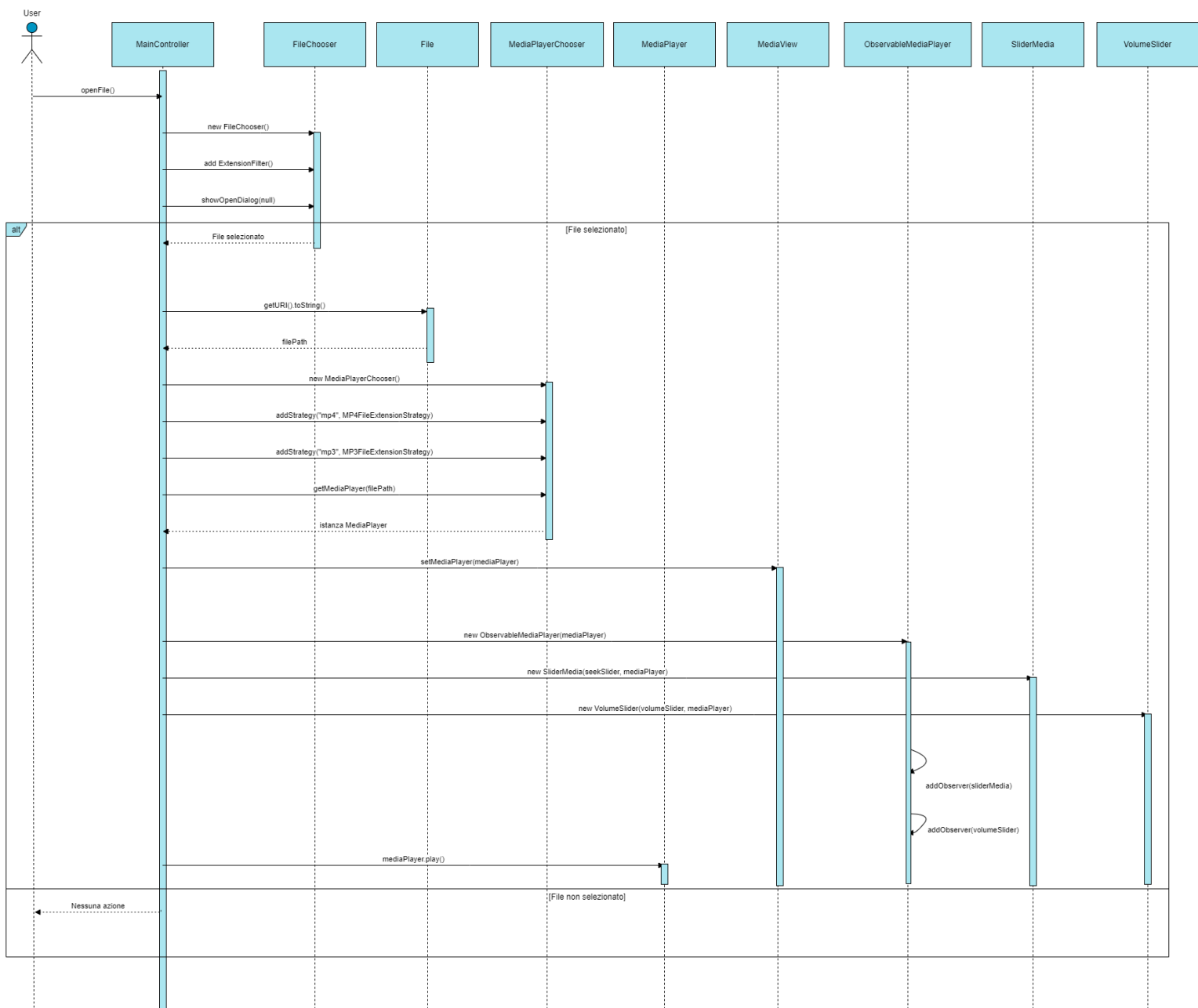
4.2.1 - Descrizione class diagram:

La classe principale di questa applicazione è il MainController, il cui compito è gestire ogni azione possibile da parte dell'utente, attraverso metodi specifici. Inoltre possiede istanze di altre classi quali MediaPlayerChooser e ObservableMediaPlayer che consentono l'utilizzo di design pattern per poter estendere in modo semplice ed efficace l'applicazione. In particolare la classe MediaPlayerChooser permette di aggiungere una particolare estensione di File e di conseguenza creare il relativo MediaPlayer seguendo lo Strategy Design Pattern e l'Abstract Factory Design Pattern. Invece la classe ObservableMediaPlayer viene utilizzata per poter rendere osservabile, il relativo MediaPlayer da classi come SliderMedia e VolumeSlider, che si occupano rispettivamente dell'aggiornamento dello slider della riproduzione e di quello del volume. Come ormai intuito, l'intera progettazione del class diagram fa uso di design pattern per rendere espandibile e modificabile il progetto in futuro. Ogni design pattern verrà descritto meglio nella sezione 5 del documento.

4.3 - Sequence diagram:

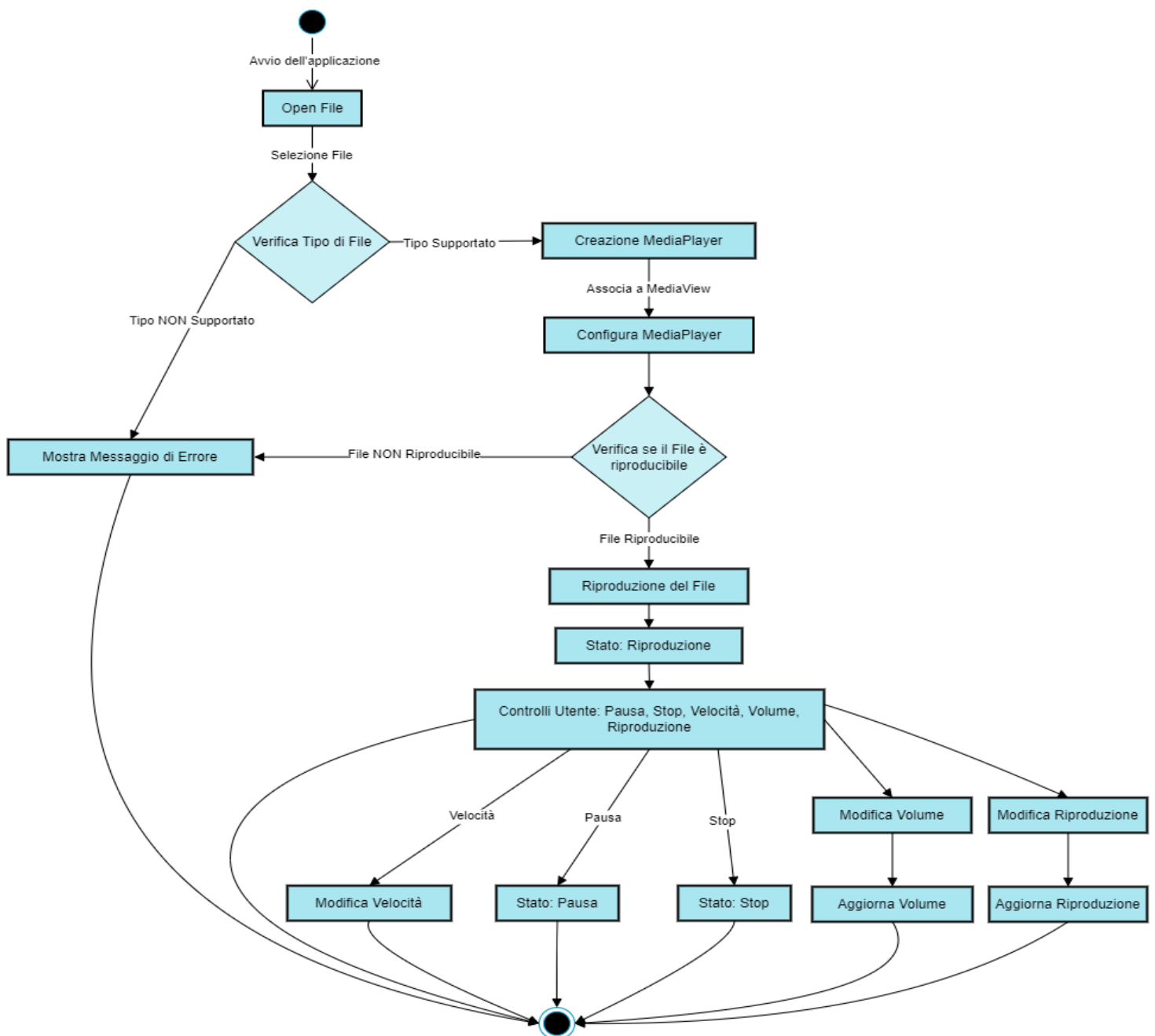
Di seguito viene riportato il sequence diagram rappresentante gli oggetti e le relative azioni compiute, quando l'utente decide di aprire un file da riprodurre. Per poter visualizzare al meglio il contenuto è consigliabile seguire il seguente link:

[sequence-diagram.pdf](#)



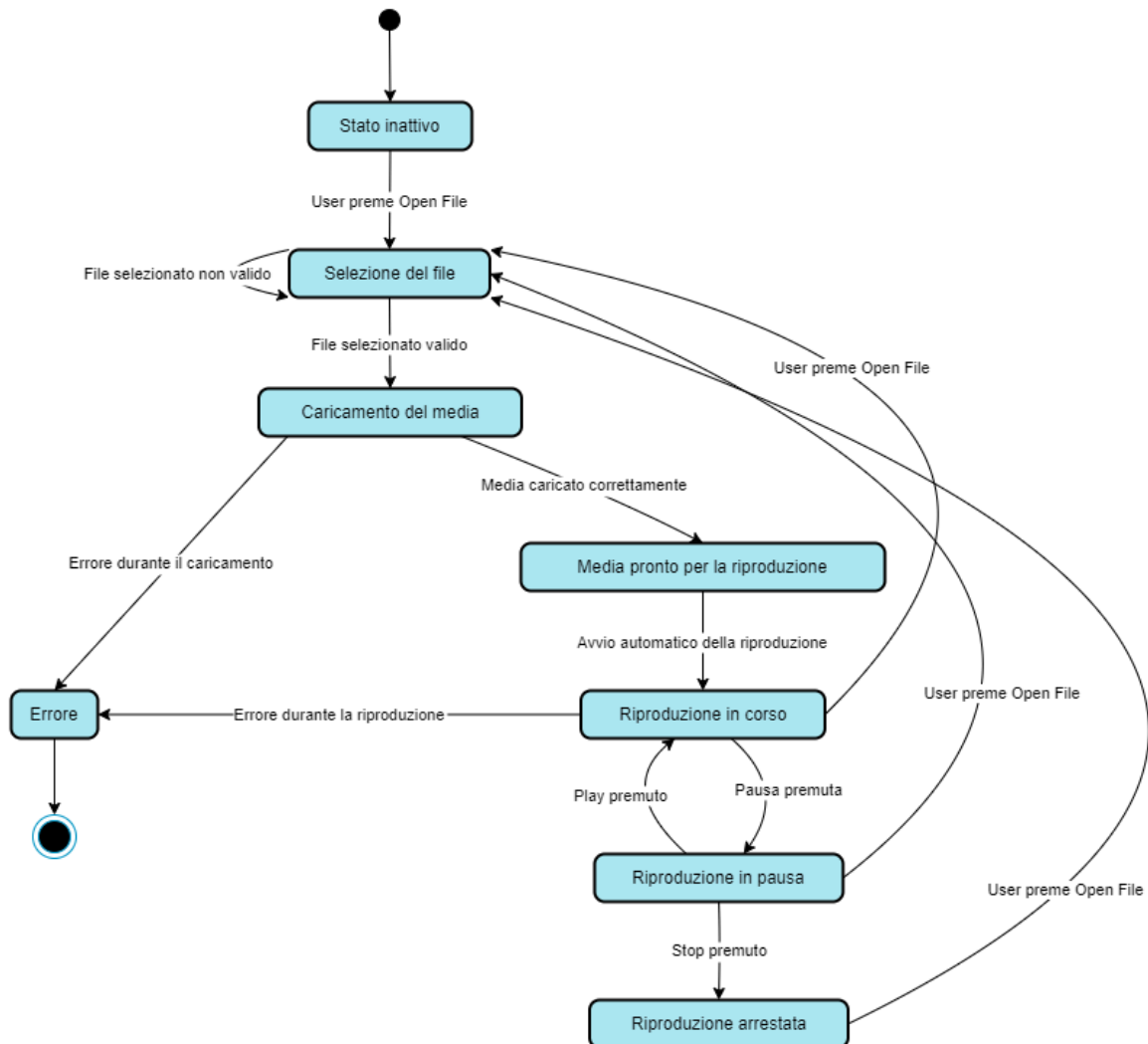
4.4 - Activity diagram:

Di seguito viene mostrato l'activity diagram relativo a tutte le possibili azioni e controlli che vengono effettuati dal sistema e dall'utente, durante l'utilizzo dell'applicazione:



4.5 - State diagram:

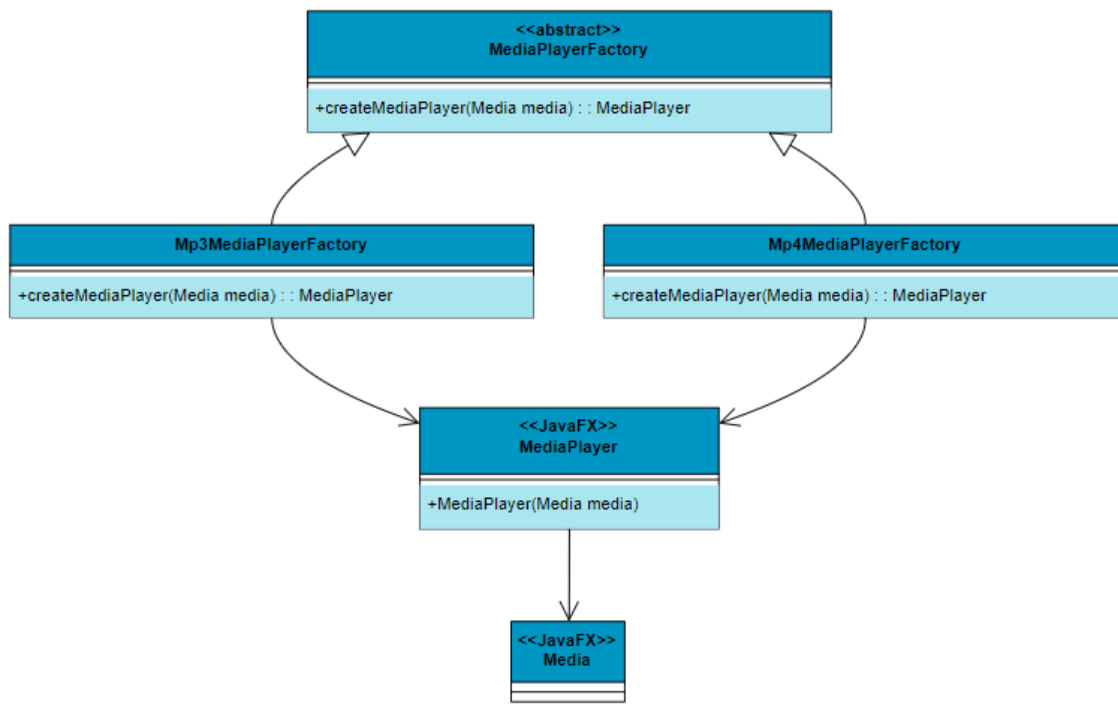
Di seguito viene mostrato lo state diagram rappresentante tutti i principali stati in cui si trova l'applicazione durante il suo utilizzo:



Nota: Non sono state inserite le transizioni interne per la modifica del volume e della riproduzione stessa, sugli stati "Riproduzione in corso", "Riproduzione in pausa" e "Riproduzione arrestata", per non appesantire ulteriormente il diagramma e perchè non si ritenevano di rilevante valore per il contesto raccontato da questo tipo di diagramma stesso, dato che si voleva mettere in evidenza i principali stati e le proprie transizioni esterne.

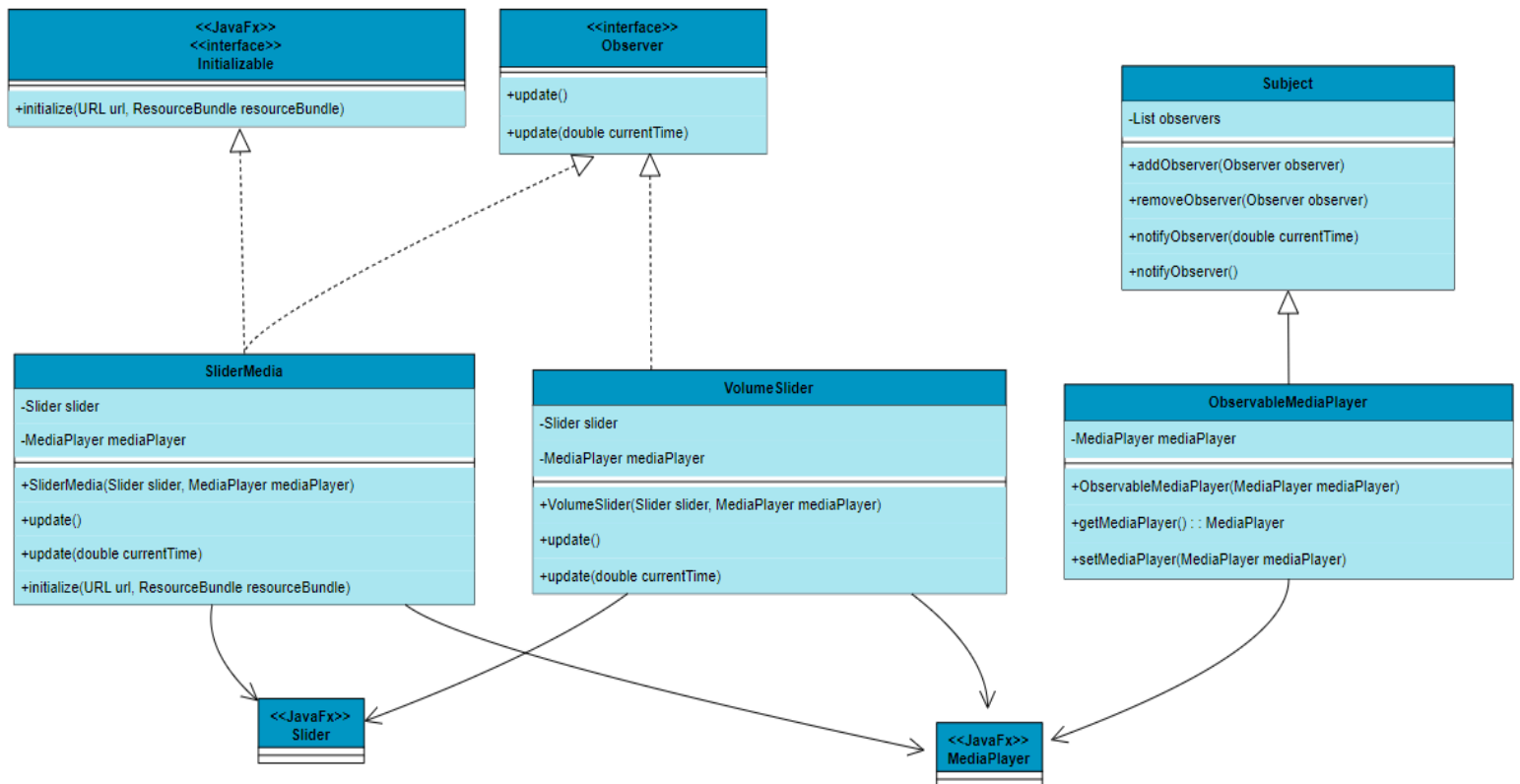
5 - Design pattern:

5.1 - Abstract Factory di MediaPlayer:



Per la creazione dei MediaPlayer è stato deciso di utilizzare l'Abstract Factory Design Pattern, in modo tale da rendere facilmente estendibile l'applicazione nel caso di aggiunta di nuovi tipi di estensioni di file da riprodurre e in generale separare il comportamento di MediaPlayer dalla sua creazione. Infatti nel caso si dovesse scegliere di aggiungere per esempio l'estensione file video AVI, basterebbe creare una nuova classe `AviMediaPlayerFactory` a cui delegare il compito di creare un nuovo MediaPlayer relativo all'estensione AVI (vedi sezione 5.3).

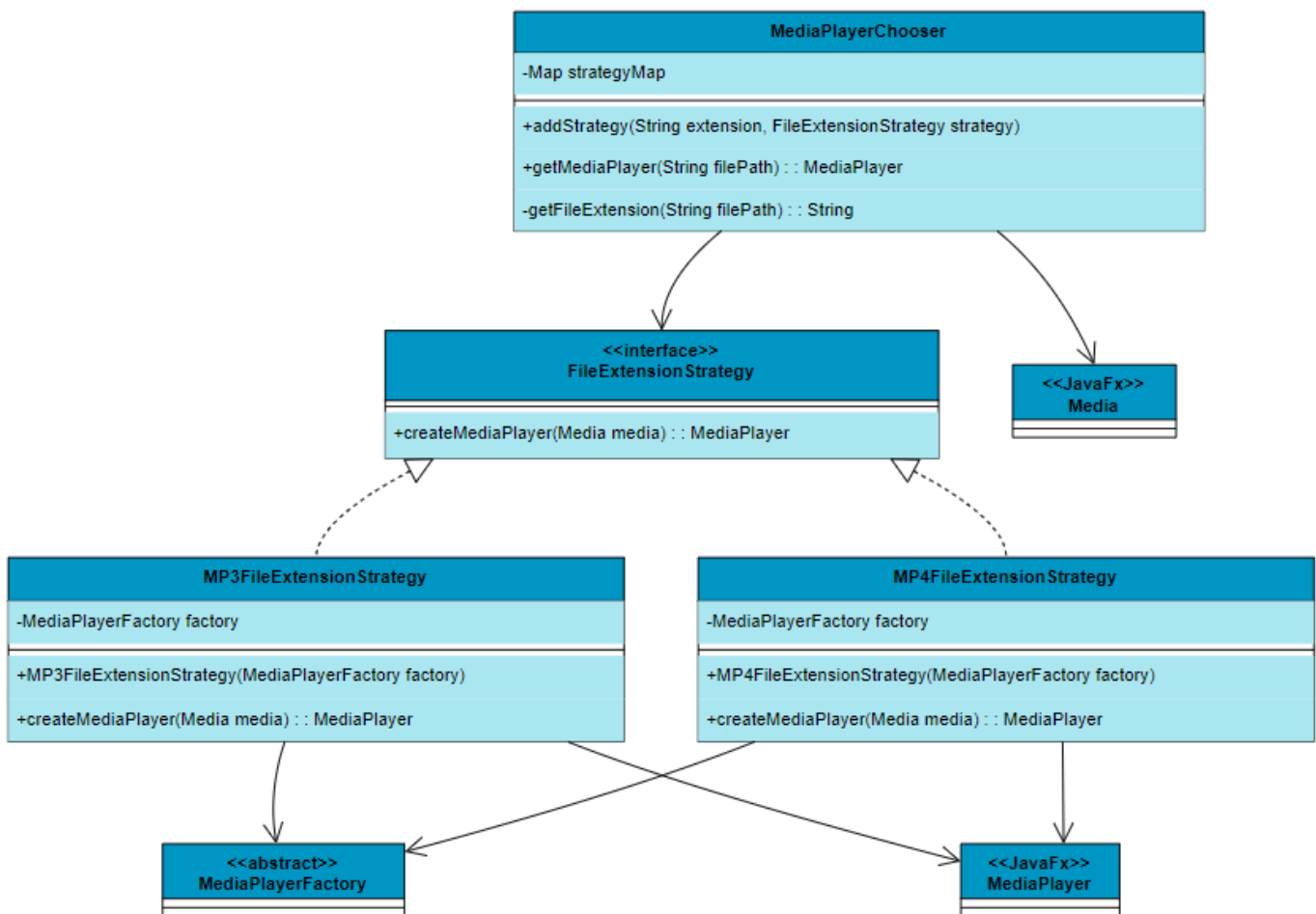
5.2 - Observer per ObservableMediaPlayer:



Per la gestione di comportamenti dinamici e di aggiornamento in tempo reale di componenti grafiche dell'applicazione si è deciso di utilizzare l'Observer Design Pattern. Come si evince dal diagramma soprastante si è deciso di creare una classe, **ObservableMediaPlayer** che estende **Subject**, a cui possa agganciare oggetti per aggiornamenti di qualunque tipo rispetto a **MediaPlayer**. In questo caso a **ObservableMediaPlayer** vengono aggiunti due observer: **SliderMedia** e **VolumeSlider**. Queste due classi sono responsabili, rispettivamente, per gli aggiornamenti sulla riproduzione e sul volume del **MediaPlayer**.

In generale questo pattern ci permetterà anche in futuro di aggiungere facilmente oggetti sempre più complessi, per permettere comportamenti dinamici al nostro **MediaPlayer**.

5.3 - Strategy per le estensioni dei file:



Per la gestione delle estensioni di file che l'applicazione può gestire si è deciso di utilizzare uno Strategy Design Pattern. Questo rende in futuro possibile l'aggiunta di ulteriori estensioni senza modificare di fatto la logica del programma. Infatti si potrebbe direttamente creare una nuova classe relativa ad una nuova estensione di file, a cui poi legare un nuovo MediaPlayerFactory e di conseguenza un relativo MediaPlayer.

La classe MediaPlayerChooser, come già detto in precedenza, è utilizzata per mantenere una mappa delle strategies, che vengono utilizzate per creare in modo corretto il relativo MediaPlayer, in base all'estensione del file che viene selezionato dall'utente.