oggetto	Relazione progetto Programmazione a Oggetti
gruppo	Ceron Tommaso, mat. 2101045 Parolin Dennis, mat. ######
titolo	MediaLibrary

Introduzione

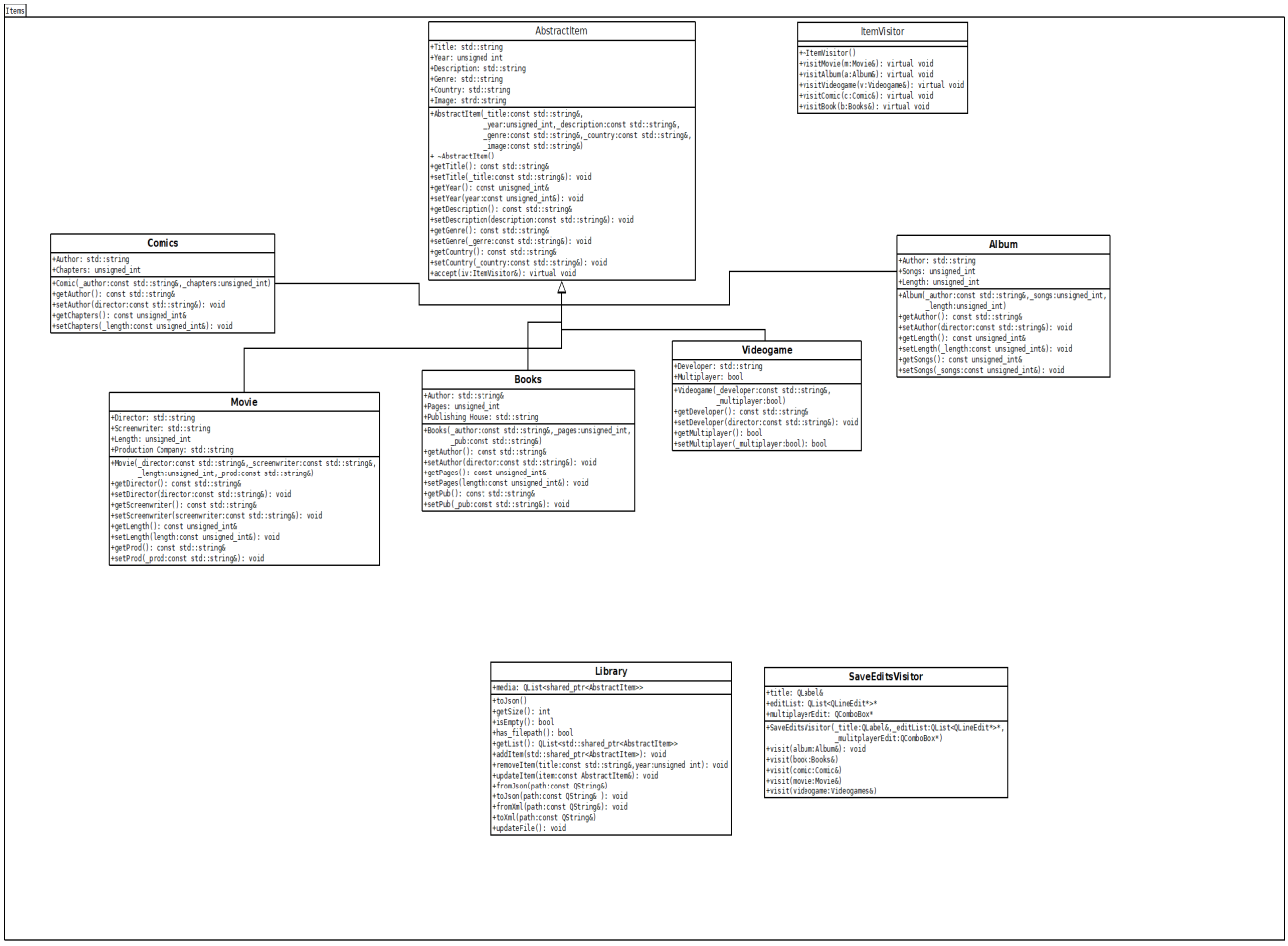
MediaLibrary è un gestore di contenuti multimediali, nello specifico Album musicali, libri, fumetti, film e videogiochi. Il programma permette la creazione, l'eliminazione, la modifica e la visualizzazione di questi oggetti. La visualizzazione può essere sia “a griglia”, ovvero più oggetti contemporaneamente, oppure “dettagliata, dove si mostrano le informazioni specifiche di un determinato oggetto.

Il nucleo del programma è rappresentato dalla classe Library, che permette la gestione dei singoli oggetti multimediali tramite una lista, e mette a disposizione tutti i metodi necessari per effettuare le operazioni descritte sopra. Gestisce inoltre la persistenza dei dati, tramite file XML e JSON.

Descrizione del modello

Il modello logico si articola in 3 parti: gli oggetti, la libreria e la gestione dei file XML/JSON.

La parte relativa agli oggetti e alla libreria è rappresentata dal diagramma allegato, la gestione dei file XML e JSON è delegata a classi di utilità che si occupano del parsing e della scrittura su file. Per facilitare queste operazioni abbiamo utilizzato alcune classi di Qt come ad esempio QJsonObject e QxmlStreamReader/Writer.



Il modello parte dalla classe base AbstractItem, che viene implementata dalle 5 classi derivate relative ai 5 tipi di oggetto multimediale. Ognuna di queste implementa il metodo virtuale *visit*.

Poichè le classi contengono solamente informazioni e non presentano funzionalità particolari, tutte le operazioni che li coinvolgono vengono implementate dal pattern Visitor, che effettua operazioni specifiche a seconda del tipo dell'oggetto. Ad esempio SaveEditsVisitor si occupa di salvare le nuove informazioni di un oggetto.

Anche per la gestione dei file JSON/XML utilizziamo il pattern Visitor, in particolare per la scrittura: il visitor infatti si occupa di creare un oggetto XML/JSON e di aggiungere esclusivamente i campi dati relativi a quello specifico tipo di oggetto.

Polimorfismo

Il principale utlizzo del polimorfismo si può ritrovare nel pattern Visitor: nel codice del programma viene infatti utilizzato più volte: per gestire la scrittura su file Json/Xml, per gestire il salvataggio di un nuovo oggetto e per mostrare la scheda dettagliata di un oggetto.

In particolare, SaveEditsVisitor utilizza semplicemente I metodi *set* di un oggetto per impostarne Il suo contenuto a seconda del tipo di quest'ultimo.

Nell'interfaccia grafica invece abbiamo utilizzato Il pattern Visitor per mostrare le informazioni dettagliate di un determinato oggetto selezionato dall'utente: per fare ciò ovviamente dovevamo avere una scheda che mostrasse gli attributi corretti a seconda del tipo dell'oggetto. Abbiamo dunque creato un *ItemDetailVisitor* che si occupa di creare una interfaccia “standard” comune a tutti gli oggetti, aggiungendo le coppie Qlabel/QLineEdit specifiche per ogni tipo di oggetto. In particolare l'attributo *Multiplayer*, che indica la presenza o meno della modalità multigiocatore, è presente sono nel tipo di oggetto *Videogame* e richiede una classe QcomboBox piuttosto di una QlineEdit. Inoltre, l'attritbuto *Length*, comune a tutti gli oggetti tranne a Videogame, si misura in unità diverse: minuti per Album e Film, numero di pagine per I libri, e numero di capitoli per I fumetti. Abbiamo quindi creato una classe LengthEdit, che eredita da QlineEdit, e che funziona allo stesso modo di una QlineEdit standard, ma mostra l'unità di misura corrispondente al tipo dell'oggetto e permette esclusivamente l'inserimento di numeri, in un certo range (ad esempio impedisce l'uso di numeri negativi).

Infine, abbiamo utilizzato il pattern Controller per creare la classe *ItemController*, che si occupa di creare un oggetto specifico, con I campi dati passati dalle classe QlineEdit della interfaccia grafica, e di mandarlo alla libreria.

Persistenza dei dati

Per la persistenza dei dati viene utilizzato il formato JSON e XML, entrambi hanno caratteristiche simili e incapsulano la libreria in un array di oggetti, composti da associazioni chiave-valore relative ai campi dati degli oggetti.

Funzionalità implementate

Funzionalità logiche

- Gestione di cinque categorie di oggetti multimediali
- Scrittura e lettura su formato JSON/XML
- Barra di ricerca e filtraggio per categoria

Funzionalità grafiche:

- Barra del menù con funzioni: salva su file, apri file e esci
- Menù laterale con barra di ricerca e filtro categorie, con relative icone
- Bottoni per operazioni CRUD
- Scorciatoie da tastiera
- Visualizzazione a griglia e dettagliata
- Utilizzo di un ItemDelegate per la locandina e il titolo di un oggetto

L'interfaccia grafica fa largo uso della struttura Model-View che consente, anche all'interno dell'interfaccia grafica, di separare il modello logico dalla parte di visualizzazione. Il *model* si occupa di prelevare i dati degli oggetti della libreria, effettuare eventuali operazioni su di essi (come ad esempio il ridimensionamento delle locandine), e mandare i dati alla *view*, che si occupa quindi solamente della visualizzazione. In questo modo la view non si preoccupa mai dello stato effettivo della libreria o degli oggetti, ma si limita a visualizzarne i dati forniti dal modello. In questo modo si semplificano anche le operazioni di aggiornamento, poiché quando la libreria verrà aggiornata invierà un segnale al modello, che provvederà automaticamente ad aggiornare la view.

Infine, utilizzare questo approccio permette molte operazioni interessanti sui dati degli oggetti: nel nostro caso le immagini puntate dal campo *image* dell'oggetto vengono ridimensionate e viene utilizzata la classe QItemDelegate nella visualizzazione a griglia, per mostrare la locandina e il titolo in sovraimpressione.

Rendicontazione ore

Attività	Ore Previste	Ore Effettive
Studio e progettazione	5	7
Sviluppo del codice del modello	15	15
Studio del framework Qt	5	10
Sviluppo del codice della GUI	10	20
Test e debug	5	5
Stesura della relazione	2	5
totale	42	**62**

Le ore sono state superate soprattutto a causa dello sviluppo dell'interfaccia grafica, che ha richiesto uno studio più approfondito del framework Qt e di alcuni pattern.